EXPERT SYSTEM DESIGN AND IMPLEMENTATION FOR MULTICHANNEL SLEEP EEG SIGNAL PROCESSING

By

CHEOUNG NAM LEE

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1985

To my parents

for their love, support and encouragement

To my friendly adviser Dr. Jack R. Smith

for sharing all his knowledge of life

To my wife and children

for their love, love and love

ACKNOWLEDGEMENTS

The author would like to thank his supervisory committee for their aid and guidance. The author is most grateful to Dr. Jack R. Smith, whose support and willingness to share every bit of his knowledge have been a source of strength during this study.

He is also thankful to fellow graduate students Sunil Mehta, Chong T. Kim, and Tai G. Chang for their encouragement and moral support.

The author sincerely thanks his parents and his family for their support and confidence, especially his wife for her endless patience and love.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

AN EXPERT SYSTEM DESIGN AND IMPLEMENTATION FOR MULTICHANNEL SLEEP EEG SIGNAL PROCESSING

BY

CHEOUNG NAM LEE

MAY 1985

Chairman: Professor Jack R. Smith
Cochairman: Professor Donald G. Childers
Major Department: Electrical Engineering

The analysis of the sleep record is a required process in sleep research. Usually, several EEG (electroencephalogram) channels, and one or two EOG (electrooculogram) channels are recorded simultaneously on the paper during the sleep period. The record of a single night's sleep data is about 500 meters long. In order to provide a fast and an objective method, many computerized sleep EEG systems have been developed. Though several systems showed good performance, there still exist problems of a gestalt perception of the waveform and a data analyzing method, which is needed to utilize the various relationships of the data. This dissertation describes a method of sleep EEG data analysis. An expert system is designed to provide the method which can simulate the signal understanding procedures of the human expert.

In this system, the EEG signal of the specified waveforms is regarded as a language, which has a set of words and a simple grammar. Each waveform is related to a word, and channel relationships or waveform relationships in time constitute the grammar. Using these words and grammar, the context analysis is done on the signal data. Then the system synthesizes all channel activities and makes a synthetic report using expert knowledge base. A mixed technique of a rule-based system and a frame scheme is utilized for the system reasoning. The frontend processor for language token generation is implemented using an A/D converter and a TM-990/101M microprocomputer having 32K bytes memory. The main processor for the language token processing is implemented on a PDP-11/23+ microcomputer with the UNIX operating system. The rapid eye movement (REM) period detection and the sleep staging which utilize multichannel information and the expert knowledge base are shown as components of the expert system.

CHAPTER 1
INTRODUCTION

Analysis of the sleep EEG record is a productive method in the study of sleep. Usually, several EEG (electroencephalogram) channels, and one or two EOG (electrooculogram) channels are recorded simultaneously on the paper during a sleep period. The record of a single night's sleep data is about 500 meters long. In order to provide a fast and an objective method for sleep EEG analysis, computerized sleep EEG systems [Ma72] [Fr73] [Ga73] [Sm78] [Ha83a] have been developed over a period of years. Although several systems performed well, today there still exist significant problems of visual perception and contextual understanding of the waveforms in multichannel sleep EEG analysis using a computer. A data analyzing method is needed, which mimics the human perception procedures and which includes more of the corresponding expert's knowledge base.

A perceptual understanding and a qualitative analysis of recorded signal data need various kinds of knowledge bases. Most of the data analysis needs information transformation which uses different knowledge bases depending on the required analysis. In particular, when the data analysis includes a visual perception process and a contextual understanding simultaneously, an expert knowledge base is necessarily required to understand and analyze the data.

In other words, in a human perception process, a picture can be regarded as an integrated knowledge source that can transfer any kind of information depending on the situation, location and the people who are looking at it. The image of the picture may stimulate a memory about a similar scene or related people, or cause us to think about implied ideas which the picture might suggest. However, recognition processes of the picture, i.e., information transformation from picture to us, are performed using a knowledge base which has been built through our daily life. Depending on the knowledge base, a different analysis result can be obtained with the same picture. The necessary information may be obtained directly from the picture and indirectly from the embedded knowledge structure of the picture using our knowledge bases. The understanding procedures of the sleep EEG record are the same as the picture understanding procedures except for the fact that a sleep data record and a heuristic knowledge base of human EEGer (electroencephalographer) are used.

Let's take a close look at the sleep data and data processing mechanism of a human expert to see why most of the conventional methods [Ha83a] [Ge80] [La80] [Sm78] do not work properly and how the data processing mechanism can be realized appropriately using a microcomputer. Fig. 1.1, Fig. 1.2, Fig. 1.3 and Fig. 1.4 show examples of sleep stage 1 REM (rapid eye movement), eye movement in sleep stage 0, K-complex in sleep stage 2, and delta activities in sleep

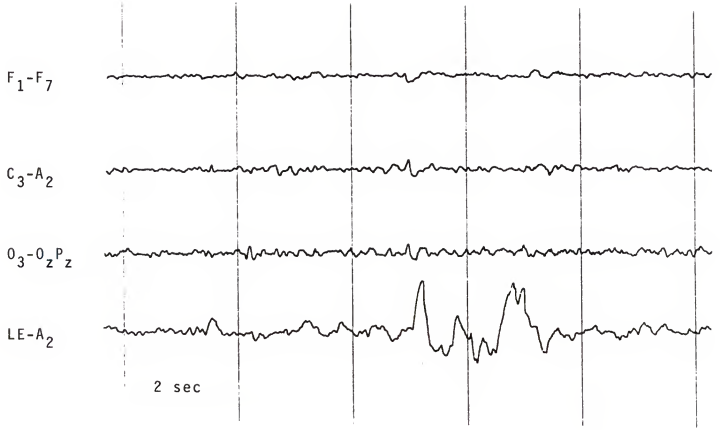


Fig. 1.1 Sleep Stage 1

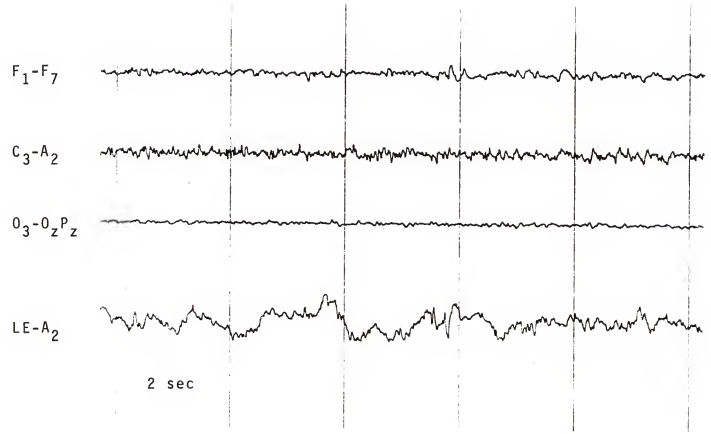


Fig. 1.2 Sleep Stage 0

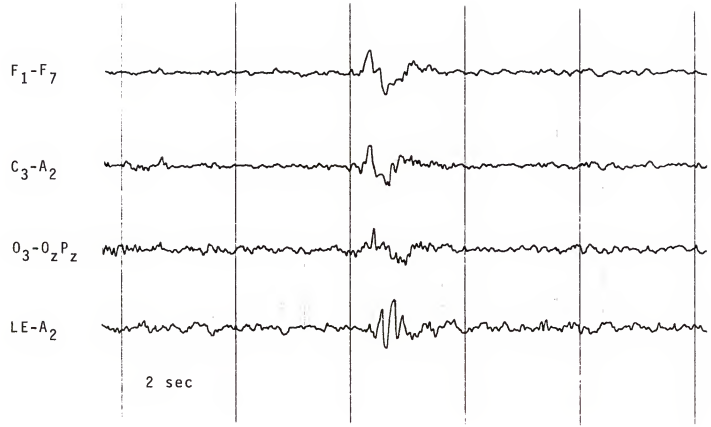


Fig. 1.3 Sleep Stage 2

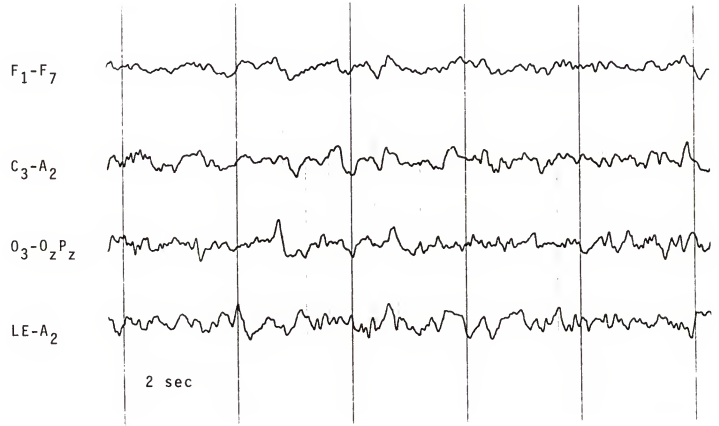


Fig. 1.4 Sleep Stage 4

stage 4, respectively. Although slow waves which look like a rapid eye movement appeared on each EOG channel (LE-A2), only the slow wave of the Fig. 1.1 is regarded as a REM period in sleep stage 1. A big slow wave on EOG channel is not regarded as a REM in Fig. 1.2 because of the dominant alpha activity in the occipital channel (O3-OzPz) in the previous seconds In Fig. 1.3, though a big slow wave on EOG channel appeared without any dominant activities in previous seconds, the simultaneous slow activities followed by a fast activity in the occipital channel makes it a K-complex wave instead of REM. In Fig. 1.4 the delta activity in all channels shows that the EOG channel slow waves are not REM. These are good examples of how the waves are recognized for REM period classification.

The data analysis mechanism can be grossly described as a two-step procedure. The first step is the visual perception of the waveform such as alpha, beta, delta, sleep spindle, etc.. The second one is the resolution of the significance of the waveforms, i.e., contextual understanding of the waveforms such as waveform priorities or total activity time of a specified waveform. Although the waveforms are classified depending on the frequency range, it is not clear enough which frequency range a certain waveform belongs to. The reasons are the following:

a) The term "frequency" itself is not clearly defined since sleep waveforms are usually composed of many different frequencies. The EEGer counts the number of

peaks or the number of zero crossing points in a fixed interval instead of decomposing the waveforms into sinusoidal waves for frequency measurement.

b) Depending on the context of data, only a certain frequency component is counted in visual analysis.

c) Each waveform amplitude is subject dependent, and all the waveforms in one subject are not always big or small. For example, a subject with big alpha wave does not always have big beta, delta, or sigma wave. Therefore, an amplitude threshold value can be changed for a certain waveform detection.

It is difficult even for a human expert to say which component is the dominant one in many instances. Also, the boundary of a waveform itself often is not clearly defined. That is, most of the waveforms have fuzzy shapes and fuzzy boundaries. These are the principal reasons which account for the disagreement of the analysis results between different human experts and why many sleep laboratories use different sleep staging rules.

However, depending on the number of recognized waveforms or the duration of a wave burst or wave activity, the contextual understanding of each waveform is performed. Usually, the context understanding mechanism for these kinds of fuzzy signals is difficult to implement using conventional programming techniques. When a visual perception [Ri83] [Wi79] [To74] is required for data analysis, specific methods or algorithms must be used depending on the problem.

These kinds of methods or algorithms cannot be generalized as the human expert knowledge cannot be generalized. Generally, different domains of expertise have different knowledge bases.

Then how can we make a microcomputer interpret the sleep data as does a human expert? To make a computer interpret a picture (data) means not only giving all the information about the picture (data) itself but also giving a meta knowledge which can operate on a given knowledge base to extract the necessary information from the picture (data). Most of the work in the field of Artificial Intelligence has centered around the problem of understanding in cognitive science, i.e., structured knowledge representation for information storage and retrieval, language understanding for communication and knowledge base management for learning i.e. knowledge accumulation. Techniques already developed to impart intelligence to a computer have been utilized in many fields such as CAD/CAM (computer aided design, and manufacturing), Signal Processing, Expert Systems, Data Base Management, etc..

To analyze the sleep EEG data using a microcomputer, an expert system has been constructed based on the human visual perception and contextual understanding mechanism which utilize the waveform detection method [Sm78] and symbolic signal processing techniques of Artificial Intelligence. The system performs:

a) the recognition of the specified waveforms in time

domain,

b) the examination of the data back and forth in time and across all channels, if necessary (qualitative analysis),

c) the quantification of the data, i.e., summation of the total activity time or the total number of bursts in each epoch (quantitative analysis),

d) the rule selection, which is dependent on the required analysis (system flexibility),

e) the rule management, i.e., knowledge integration (learning mechanism),

f) an explanation of the reasoning mechanism (user friendliness).

At present, the user friendly interface (such as natural language communication or data display) and knowledge base management system have not been fully considered. Only the basic frame of the sleep EEG analyzing expert system has been designed to demonstrate how it can simulate a human expert. In the present system, a data analysis report more detailed than the one provided by the human expert is generated using the multichannel EEG/EOG signal inputs. The symbolic signal processing is performed using a simple signal language understanding mechanism and the heuristic rules of the human expert.

The functional structure of the data analyzing system is divided into two blocks, a token generator and a token processor. In the token generator, the multichannel EEG

signals are converted to a token stream which contains the information about all channel waveform activities. From the token stream, a sequence of words is generated, which describes a channel's waveform activity. This sequence of words is interpreted in the token processor. The token processor's interpreting mechanism is very similar to that of high level computer language compiler. A set of words is regarded as a signal interpretation language. Each waveform is related to a word, and the channel relationships or the time domain waveform relationships constitute the grammar of the signal interpretation language. Using the selected words and grammar, the context understanding of the signal is performed. Although all the signal characteristics are not represented in the grammar, basic signal patterns and relationships are recognized and analyzed, using a reasonably simple grammar. Once a channel description list is obtained, an epoch description is obtained using a knowledge base of the heuristic rules of the human expert. A hybrid technique of a rule based system and a frame scheme is utilized for the rule implementation and the system reasoning.

The token generator, used as a frontend processor, is implemented using TI-9900 microprocessor and an A/D converter in which four single end input channels are utilized. The token processor, used as a main processor, is currently implemented on a GOULD CONCEPT 32/8280 minicomputer using the UNIX operating system. (In the near future it will be implemented on PDP-11/23+). The TI assembly language, and the

high level language "C", and "LISP" are used in the token generator and processor. The standard software of the UNIX operating system, the LEX (lexical analyzer) [Le75] and the YACC (yet another compiler for compiler) [Jo75] program, are utilized for generating the signal language compiler.

In Chapter 2, most of EEG analysis methodologies which are related to sleep EEG analysis are reviewed. In Chapter 3 a more elaborate approach to a sleep EEG analysis expert system is discussed. In Chapters 4 and 5, the design concept, the detailed structure of the system and the system status are described. The system performance and the conclusions are discussed in Chapter 6.

CHAPTER 2
OVERVIEW OF EEG SIGNAL ANALYSIS

In this chapter the EEG signal source, model and components are briefly reviewed and the basic EEG analysis methods are discussed to present a better picture of the advantages and limitations of each method for the sleep EEG analysis application.

EEG Signal Source, Model and Components

It is generally known that the scalp EEG derives from graded synaptic potentials generated by pyramidal cells in the cerebral cortex, which are triggered by rhythmic discharges from the thalamic nuclei. A pacemaker system situated in the thalamus and in the reticular formation probably regulates the synchrony of the cortical signals [An68] [Ge75]. The relatively slow time course of EPSPs (excitatory post-synaptic potentials) and IPSPs (inhibitory post-synaptic potentials) is comparable with the EEG and the summation is facilitated by the columnar structure of the neurons reaching from upper to lower layers of the cortex [Co80].

Several models have been devised to locate the signal source, to characterize the wave propagation, and to analyze the signal characteristics. One of them is the current dipole layers model [Nu81a] which assumes that the scalp potentials are due to current dipole layers occupying various

surface areas of the cortex. Approximately 2/3 to 3/4 of cortical neurons are oriented perpendicular to the cortical surface. The large number of interconnections makes synchronous activity possible. Using this current dipole model, a relationship between cortical current density and resulting scalp potential was obtained by means of arguments about volume conduction through brain, skull, and scalp. And a theoretical model of neural interaction at the macroscopic level of brain hierarchy was developed to predict cortical current density and the resulting spatial-temporal properties of scalp EEG. It was noted that the EEG frequency and amplitude is dependent on the threshold value which is the average difference between the number of excitatory and inhibitory synapses on a neuron, weighted by the appropriate threshold [Nu81a].

A neuronal population model [Ch73], which is based on the hypothesis that the surface potentials are a combination of ESPSs and IPSPs which occur both at different depth and different latency, was used to quantify the electrogenesis of the evoked surface potentials and to examine the relationship between scalp potentials evoked by visual stimulation and neuronal functioning. This model is used to determine the location (depth) of the signal sources and the approximate size of the neuronal population responsible for the potential. Using an array of scalp electrodes over the occipital cortex and computer simulated experimental data, it was shown that the sequence of neuronal excitation, the

magnitude of excitatory current, the depth of the neuronal population, and the period of activity of the EPSP and IPSP could be determined [Ha75] [Ch77].

For the quantitative analysis of the EEG signal, a hierarchical model [Sa80] which utilizes a recursively estimated autoregressive (AR) model, a piecewise stationary model, composite source model and character string and syntactic model, was developed considering that EEG signals may appear to be stationary over short periods of observation, but exhibit markedly nonstationary behavior when observed for a long period. In this model a discrete time signal model of a piecewise stationary stochastic sequence is used for data segmentation. Each AR model for a data segment represents a type of waveform. The transition of the activity is detected using a one-step prediction error. Using this segmented or piecewise stationary model, the decomposition of the original EEG signal is done to produce a composite EEG signal source. This composite source generates the output of character strings by random switching among the subsources. The output of the composite source is then analyzed using a stochastic grammar in which the detected substrings are regarded as a nonterminals. To remove the subject dependency of the model, a global subsource set for the given population was developed. With a ninety minutes data record, about 800 segments and 46 identified source models were obtained at a speed of five to ten times real time processing of a single EEG channel.

Although certain models for EEG signal sources could locate the spatial origin and could explain wave propagation characteristics and wave dispersion relationships in a nonhomogeneous volume conduction such as the brain, cerebral cortex, skull and scalp, there is not sufficient knowledge of the EEG signal origin and the significance of EEG activities such as the relationships between physiological and clinical states. However, the analysis of the EEG record is a useful aid in the research of the clinical diagnosis, the evoked response potential and the sleep analysis etc. [Ge75] [Re77] [Ba79] [Ge80] [Dz84] [Bo80] [Az82].

Two kind of signals are of main concern in brain wave research. The first signal is the event related potentials (ERP). External inputs to the human brain such as sensory inputs generate these potentials. For instance, the N100 component [Co80], which has about 100 ms latency, can easily be evoked by auditory, visual and tactile stimuli. It can be seen in a central channel EEG if a simple stimulus, such as a hand clap, is given. The amplitude of this component is dependent on a variety factors including intensity of the stimulus, its expectedness and attention paid to it. Usually, event related potentials are masked by the background noise which normally exists in the low frequency band. To extract the ERP from the ongoing activity which seems like background EEG or noise, special averaging techniques such as cross-correlation, median representative, synchronous and latency-corrected averaging have been used [We72] [Mc77]

[Vi77][Au81]. The second signal is the clinical EEG signal. The clinical EEG signal can be distinguished by physically looking at the record because the signal amplitude is generally much larger than the background activities. Physical interpretation and clinical interpretation are possible by visual analysis of the clinical EEG. For physical interpretation it is necessary to recognize and classify specified EEG waveforms which have an individual identity and temporal patterns which result when they occur in a sequence. For clinical interpretation the information is obtained from the comparison of empirical observations between the EEG and clinical observations. The sleep EEG is one of the clinical EEG signals. The computerized physical interpretation of the clinical EEG data is the main concern in this study.

The signal used in sleep research is in the frequency range of 0.5 Hz to 120 Hz. The specific frequency bands used in the present system include : rslow (0.3-6.0), oslow, pslow, qslow (0.3-3.0), delta (0.5-2.0), theta (3.0-7.0), alpha (8.0-12.0), sigma spindle (11.75-16.0), beta (16.0-32.0), and artifact (32.0-120). Although waveforms are classified in terms of frequencies, the number of local peaks in a fixed time interval or the half wave duration is utilized instead of sinusoidal wave number for frequency measurement in visual sleep EEG analysis. There are slight differences in the range of each frequency band depending on the laboratory standards.

EEG Analysis Methodology

A variety of methods have been tried to analyze EEG data [Ge75] [Re77] [Ba79]. They can be roughly divided into two groups. One is the frequency domain analysis in which the spectral parameters obtained using various spectral estimation methods are utilized for classical pattern recognition methods. The other is time domain analysis in which the waveform descriptors such as wave period, peak amplitude, derivatives, statistical characteristics, and wave criteria are utilized to describe each waveform activity directly or indirectly. In the following section, various spectral estimation methods, frequency domain methods, time domain methods, and pattern recognition methods for clinical EEG classification are reviewed.

EEG Spectral Estimation

An appropriate spectral estimation is the key to the spectral analysis. In general, for a consistent spectral estimation it is assumed that the signal is stationary or at least piecewise stationary [Op75]. Though piecewise stationarity of the EEG [Co77] [Ka73] and Gaussian distribution behavior [El67] [Mc75] were shown in some cases, it is also shown that the signal characteristics such as stationarity and Gaussianity are changing [Ge75] depending upon the behavioral state of the subject. Therefore, the change of the state, i.e., the duration of the same state and the boundary of the state change need to be considered for a proper spectral estimation.

The power spectrum can be estimated using the indirect method [Bl58], direct method [Du73] [Ma73] or data model [Fe71] [Ze69]. The correlogram and periodogram have been used for indirect and direct method, respectively. AR (autoregressive), MA (moving average) and ARMA (autoregressive and moving average) techniques have been used to generate data models. The direct method has been widely used with the FFT (fast Fourier transform) or FWT (fast Walsh transform). Data model has been used as a minimum variance and unbiased estimator in modern spectral analysis to represent the time domain data structure. The direct method and data models related mainly with the clinical EEG analysis are discussed in the following section.

Direct Method

The power spectrum is usually obtained by taking the Fourier transform of the properly windowed data and squaring its absolute value [De73] [Ma73]. The characteristics of the spectrum obtained using the direct method are equivalent with the ones obtained from the periodogram. Although one can easily select an unbiased estimate of autocorrelation function for the calculation of the periodogram, the finite limits of the data cause a biased estimate of the power spectrum, and the variance of the estimated power spectrum does not approach zero even though the number of the data points used for the periodogram estimation approaches infinity [Op75]. Therefore, a consistent estimate of the power spectrum can be obtained from the modified periodogram. To

reduce the variance of estimates the average of the independent periodograms has been used. To increase the spectral resolution, an appropriate spectrum window is applied to the periodogram [Ba53]. But the averaging process increases the bias of the estimates and decreases the spectral resolution as the segmentation number increases in a fixed data record length. Another approach to this problem is to smooth the spectrum by applying an appropriate window function directly to the data sequence before computing the periodograms [We70]. Overlapping segmentation has also been utilized to minimize the decrease of the resolution caused by segmentation of the fixed data in the averaging process, where the modified periodograms are not independent. That is, one can only improve the variance of estimates at the expense of the spectrum resolution. Therefore, when a certain spectral estimation procedure is applied to the unknown signal, the interpretation of the obtained spectrum has to be done considering the spectral estimation procedures, i.e., windowing effects and sampling effects simultaneously.

Since the earliest EEG spectrum was obtained by manual calculation (see [Sm78]), FFT and FWT have been widely utilized for EEG spectral estimation [GE75] [Ge80] [Bo77] [Ba79] [Ye72] [B174] [La80]. The FFT [Co67], whose algorithms are based upon that of decomposing the computation of the discrete Fourier transform of a sequence of length 'N' into successively smaller discrete Fourier transforms in time or in frequency, appreciably decreases the computing time for

spectral calculation. At present, in many of the laboratories [Ba79] [Re77], multichannel FFT's are used routinely with the aid of hardware FFT processors which are dedicated to a micro or minicomputer.

To decrease the time-consuming computational procedure of the FFT, other orthogonal functions have been tried. Walsh functions [Ye72] [La76] is one of them. The fast Walsh transform (FWT) employs Walsh functions in the series approximation of the original signal. Walsh functions [Bl74] are a set of periodic orthogonal functions which are binary in nature and appear to be a squared-off version of the sinusoidal functions. These binary waveforms make the fast Walsh transform very attractive for many kinds of signal processing applications. Because only two function values, +1 and -1, are used, the need for multiplication in execution of the FWT is obviated by taking advantage of product identity [Dz84].

However, opinions on the efficacy of using the FWT in place of the FFT for power spectrum estimation are split, advocates versus critics. The debate centers on the question of whether original signal information can be obtained as much as with that of Fourier analysis. It has been claimed that Fourier analysis is relevant in sinusoidally based waveforms and the Walsh transform is more appropriate in cases where the signal contains sharp discontinuities and a limited number of amplitude levels [Ja81]. In the analysis of the truncation and roundoff errors associated with the

use of Fourier and Walsh series, it is claimed that for long samples of smooth signals, far more terms are required in the Walsh series representation and greater accuracy is required of their coefficients for a given root mean square total error. Even for discontinuous signals the Walsh series may require substantially more terms, thus counterbalancing the computational advantage of the fast Walsh transform [B174].

On the other hand it is reported that depending on the application, useful features can be obtained using Walsh transform. Though the discrimination between sleep stages by Walsh power spectra is not as good as the Fourier spectra for sinusoidal like EEG data due to diffusion throughout the frequency range, the activity present in the frequency spectra was evident in the Walsh spectra plot [Ye72]. In an EEG epoch data classification problem [La80] with predefined spectral features, features selected from the spectrum of sleep EEG data were compared to corresponding Fourier features. Each feature set was used to classify the data using a minimum-distance clustering algorithm. It was shown that the Walsh spectral features classify the data in much the same way as the Fourier spectral features. It is claimed that Walsh spectral features in place of Fourier spectral features enable one to take advantage of the vast computational superiority of the fast Walsh transform over the fast Fourier transform. In the application of EEG power spectral estimation to on-line monitoring, in which it is desired to

track or categorize a subject's state versus time, the FFT and the FWT were examined. The results shows that when the EEG is modeled as a Gaussian, wide-sense stationary random process, The FWT can provide useful insight into performance with actual EEG data, with care in interpretation [Sm81]. In the statistical evaluation of the quantitative and dynamic differences between the results of Walsh and Fourier analysis of the EEG done for the purpose of characterizing the effects of anesthesia, the efficacy of the FWT has been reconfirmed [Dz84]. However, for the most effective applications one should select a proper method of spectral estimation and the interpretation.

Data Models

If the signal is locally stationary, it may be more appropriate to derive a model of a data structure which has the same statistical properties as the signal [Ch81]. The signal is modeled as a linear combination of its past output values and present and past values of a hypothetical input to a system whose output is the given signal, i.e., a linear system output with a white noise input. Because the signal is predictable from the past and present values of inputs and outputs, increased spectrum resolution can be obtained by extrapolating the autocorrelation function beyond the data limit.

There are three basic data models, autoregressive (AR), moving average (MA), and autoregressive moving average (ARMA). The MA model and AR model are the special types of

an ARMA model. The AR model output is given as a linear combination of past outputs and present input. The MA model output is given only as a linear combination of past inputs. In the data model there are two problems in system function approximation, the determination of the order and coefficients of the system function. In general, assuming either the deterministic signal or random signal input, the method of least square error is used for the parameter evaluation of the model. Linear prediction [Ma76], maximum entropy [Ul75] and maximum likelihood [Ca67] [Ca69] methods have been used widely for the parameter evaluation. In the maximum entropy and linear prediction methods, the autocorrelation function is extrapolated in such a way that the entropy of the corresponding probability density function is maximized in each step of the extrapolation of the autocorrelation function [Bo71], and the expected mean square error between the next sample and the predicted value is minimized [Ma76]. In the maximum likelihood method the input is assumed as white Gaussian noise and output is the maximum likelihood estimate of data sequence. The relationship between maximum entropy spectra and maximum likelihood spectra was shown [Bu72] as the reciprocal of the maximum likelihood spectrum which is equal to the average of the reciprocal of the maximum entropy spectra.

A couple of criteria have been used to determine the order of the data model. It is known that if the input signal is a white Guassian noise then the error of the spectral

estimation decreases as the order of the filter increases. For practical purposes the minimum order of the AR filter is obtained using threshold test

$$1-Vp+1/Vp < e$$

where Vp is the normalized error of the pth order filter [Ma76]. Another criteria for determining the order of the filter is Final Prediction error [Ak74]. It is based on the fact that in the model-fitting multiple decision criteria has to be used instead of the hypothesis-testing procedures. It has been utilized to determine the order of the filter for the case of autoregressive signal plus noise [To75] [To76].

In a method of fitting an AR model to the EEG [Fe71] a second order AR model is initially set up. After calculating the autocorrelation function and the power spectral density of the original record and the AR model, the order of the filter is adjusted to satisfy the minimal error condition. The results show that the optimal epoch period for the EEG generator is 20 to 30 seconds. For the feature extraction from the EEG by adaptive segmentation [Bo77] [Pr77], the linear predictive filter was used to split EEG record into elementary patterns called segments and transients. Appropriate features, representing power spectra and the time structure of the signal, were then extracted and finally combined into a feature set representing the EEG as a whole. With the aid of second order differentiator the LP filters could give a useful degree of cluster separation between the

epileptic transients and background activity [Bi78b]. In the hierarchical model of the EEG signal [Sa80], a piecewise stationary autoregressive model was used for segment classification using the parameters of the AR model. Though it is generally understood that an autoregressive model provides a good way of spectral estimation when high resolution is needed with short data records [Ma76], one must be careful in choosing the order of an autoregressive model and the sampling rate for the statistical characteristic representation of the signal.

Spectral Analysis

In addition to an appropriate spectral estimation, an appropriate spectral interpretation such as dominant feature selection is the key to spectral analysis. The duration and location of the data window can cause improper results of the analysis. Adaptive segmentation has been utilized to locate the spectral window properly. Transient activity could be detected using this method. Spectral parameter analysis, compressed spectral array and spatiotemporal analysis have been utilized for proper interpretation of the EEG spectrum. The change of the estimated spectrum over time can be understood in terms of wave parameters.

Spectral Parameter Analysis

Spectral parameter analysis (SPA) is based on the assumption that the spectral parameters describing the signal do not vary during the analysis epoch which is usually 10 or 20 sec. In other words, the signal is regarded as a locally

stationary random process during such a period [Is75a] [Is75b]. SPA provides an objective description of the distribution of spectral power in the EEG signal in the form of a rationale spectrum with no more than 8 parameters. The spectrum is divided into 1-3 components that are described by frequencies and power parameters such as bandwidth, peak frequency and power. These spectral parameters are determined from the estimated power spectrum.

SPA was used for describing the normal EEG. The variations with time in the center frequency of the alpha activity and the relation between the bandwidths of delta and alpha activity were studied[We71]. SPA forms the basis of an EEG signal generator [Ze69]. The method involves determination of the parameters for a data model of the EEG signal. From the experimental data the parameters are estimated by first computing the first 100 correlation coefficients and then applying an AR or ARMA model to generate a stationary discrete sequence. The parameter estimation procedure was tried on a few records and it was found that typically a fifth order system suffices to give an accuracy of about 5 percent for the correlation coefficients which describe the stationary parts of EEG signals.

Adaptive Segmentation

Adaptive segmentation procedure is very close to quantifying the procedure of visual analysis of the EEG. This method was used for transient wave detection [Bo77] [Mi79] [Pr77]. For adaptive segmentation it is assumed that the

EEG is a piecewise stationary time series. For the feature extraction the adaptive segmentation procedure with the aid of AR model was used [Bo77]. This procedure was developed to detect the segment boundaries and locate transients, and to represent the information in the segments in terms of a set of parameters specifying their power spectra. Here, "piecewise stationary" means that within a segment, the short-time spectral estimate does not change appreciably with time. Transients are short time nonstationarities. A segment was specified by its length and power spectra estimate, a transient by a set of grapho-elements and time of occurrence. Though it showed good results for the limited data, the selection of an optimal threshold level, which is very sensitive and dependent on the subject, is not generalized.

For an adaptive segmentation of the EEG records [Pr77] from a two second data record an AR model was derived. By passing an EEG signal through this filter the spectral error is measured. When the spectral error exceeds the threshold a new AR model is derived and the transient is detected. It is reported that the threshold setting for the detection of the fast transients is more difficult and needs further validation on more material if it is to be used for clinical purposes.

Compressed Spectral Array

The compressed spectral array (CSA) is a graphical display method that can compress and simplify complex data.

This emerges from the multiple scalp electrodes over considerable intervals of time. How and in what location the EEG changes in frequency and amplitude is observed easily with the CPA, using a hidden line suppression algorithm [Fo82] and fast Fourier transform. It is displayed in terms of specttral power in the frequency range of 0 to 16Hz. CSA has been practically useful in monitoring the anesthesia level of the patients in the operating room [Bi73]. By displaying the spectra using a general computer, hemisphere asymmetries can also be detected. However, in multichannel application, because of excess data in a short period there are difficulties in the interpretation of consecutive spectral arrays.

Spatiotemporal Analysis

It is important to recognize a spatiotemporal pattern since the occurrence of EEG pattern in time and space cannot be separated. In a spatiotemporal analysis it is assumed that there are a few widely spaced EEG signal generators in the cortex that have the same frequency with different phases [Jo69]. That is, EEG of the scalp is regarded as a traveling wave . As a technique for displaying the EEG spatiotemporal characteristics, a spatiotemporal map shows how the potential distribution along the lines of electrodes changes with time [Re68]. A contour map presents the distribution of potential over an area of the head at one instant of time. It is derived from a two dimensional electrodes array. Using interpolation, points of the same potential are joined together. Spatiotemporal analysis is used

to locate the origin of abnormal EEG waves [Br75]. The multidimensional filtering which utilized the multidimensional Fourier transform, is used to decompose composite wavefronts [Ch77]. The method makes it possible to track selected wave propagations across the array of scalp electrodes.

Waveform Parameter Analysis

The information of the picture-like EEG data can be understood and transformed by visual recognition better than with any other method. Individual waveform recognition or detection is the key to waveform analysis. Once a certain waveform is recognized, it is interpreted by considering the context of the same channel or the other channels. The general characterization of an amplitude-time pattern of each epoch and the detection of the individual wave patterns which can be defined in advance are necessary for a qualitative description of the sleep EEG data. Several kinds of methods have been utilized to describe the EEG activity in time domain. Amplitude and period of the waveform are the basic parameters which are used to describe waveform activities in ways similar to the visual evaluation. The first and second order derivatives of the waveform have also been utilized to check the sharpness and the local peaks of the wave as the description parameters.

Amplitude Analysis

Each epoch's activity is represented by the total amount of the absolute amplitude of the EEG. In a sleep staging application [Ag67] [Ag73], it was reported that an

amplitude analysis of the EEG alone was not sufficient for the sleep stage determination because : the contingency between the stage of sleep and the electrogenesis level was not a consistent phenomenon, and only sleep stage 4 has a level which can be clearly distinguished from other states. That is, the electrogenesis level is subject dependent and the detection of the specified waveforms, which are not identified in the integrated EEG, is essential for the sleep staging.

Hjorth Parameter

In order to describe an EEG data quantitatively three parameters of an EEG signal known as slope descriptors were defined in the time domain. They are equivalent to the zeroth, second and fourth moments of the normalized power spectral density. These parameters are defined as follows:

a) activity, the variance of the amplitude or mean power in the epoch;

b) mobility, the average power of the normalized derivative in the epoch;

c) complexity, the average power of the normalized second derivative in the epoch.

Using these parameters the basic properties of the spectrum can be derived without any frequency analysis, thus requiring less computation time [Hj70]. It was reported [Bi78a] that spectral analysis proved more reliable than Hjorth's analysis for both discriminating and predicting the site of cerebral pathology between patients with localized pathology

and normal control. This method has not come into general use.

Period Analysis

The base line crossing (zero crossing) of the EEG and the first or second derivatives of the EEG are determined in order to select a specified frequency band activity [It69] [Ro70] [Sm74]. A high frequency component superimposed on a slow wave can be detected using this method. The relationship between frequency and amplitude gives the various parameters which describe the characteristics of the EEG [It69] [Fr69]. It was shown that counting the number of zero crossings over an epoch is effectively equivalent to the more complex procedures of computing zero, second and fourth moments of the power spectral density [Sa71]. A comparison of spectral analysis with period-amplitude analysis (which utilizes a period and absolute amplitude histogram), applied to the narrow band EEG activity [Kt81], showed that while the power spectrum efficiently quantifies the overall power trend in the EEG data, period amplitude analysis offers more resolution than the power spectrum in detecting electrographic details in amplitude and incidence within relatively narrow frequency bands.

Waveform Detection

This method [Go74] [Sm75] is based on the human visual detection of the waveform. Specific waveforms which are used for sleep analysis are detected by checking the zero crossing and peak amplitude of the EEG waveform. It consists

of analog filters, amplitude detectors, period analyzers and wave pattern recognizers. A filter is provided for the proper operation of the zero crossing detector. Fast activity superimposed on the big slow wave has to be filtered first or the zero crossing detector can give the slow wave period instead of fast wave period. The pattern recognizer selects the specific wave pattern using minimum amplitude criteria and wave criteria [Sm78]. Over 92% agreement was obtained between human and automatic detection of sleep spindle waveforms [Jo76] [Si76].

Pattern Recognition

If the actual waveform of an EEG is not required to obtain the information required from a particular measurement or experiment, mathematical pattern recognition procedures can be used to extract information. There are two different cases in pattern recognition problem problems. One is clustering and the other is a classification problem. When prior knowledge is not available, the problem is to extract certain features which are successfully able to classify the given data set into subsets for proper classification. If a prior knowledge is not available then the problem is to form clusters which share certain common characteristics [De77].

Usually in clinical EEG data analysis, a measurement data set is classified into subsets which are mutually exclusive. The pattern recognition rules or features are derived from the heuristic rules which are obtained from the observation of data set and clinical diagnoses, or from

previously selected training data set. Many features from the frequency domain and the time domain are utilized in EEG analysis. The power spectrum obtained from FFT, FWT or data models is used as basic information for feature extraction in frequency domain. Wave parameters, such as Hjorth parameters [Hj70], wave period, slope or time of occurrence are used for time domain analysis. Most of the pattern recognition techniques have been applied [Ge80] [Mc81], including various decision functions, linear classifier, quadratic classifier and syntactic methods.

A correlation technique was used for recognition of spontaneous EEG patterns in time domain. Correlations were obtained using two different moving templates [We72]. The correlation classifier and maximum likelihood quadratic classifier were compared to classify single evoked potentials from four different visual stimuli. Forward sequential feature selection was employed to reduce data dimensionality. It was noted that if the covariance matrices of the normal distributions are not equal, i.e., noise is correlated sample to sample, the quadratic discriminant function gives better performance than the linear one [Se79]. Iteratively determined discriminant functions were used in a decision tree for all night sleep staging [Ma72]. Separate systems were designed for high and low alpha subjects and heuristic rules were also included in the decision tree. Linear stepwise discriminant analysis were also applied to sleep staging [La70]. Forward sequential feature selection is also

often used to select features in a sequential manner. To date, no satisfactory techniques have been found for an optimum feature selection for the general classification problem. A multivariate parametric classifier is compared with a K-nearest neighborhood classifier [Yu80]. It was reported that the optimized K-nearest neighborhood classifier gave slightly better results than the parametric one, the over all level of success was below 70 percent.

The syntactic pattern recognition analysis was also used [Gi79]. Training of a classifier was conducted with well defined data epochs, selected by electroencephalographers. A standard token set was obtained from these epochs. Preparse contextual analysis was used to avoid ambiguities caused by the training set of broad spectral peaks. Postparse contextual analysis was also used to evaluate cross channel differences. Although this method gave over 80 percent correctness, the selection of primitives, and production rules which govern the merging of primitives were not generalized.

Each method has advantages and limitations in application to sleep EEG analysis. In spectral analysis, the time domain information such as the duration of specific wave bursts or the number of the occurrences, which is essential for the clinical EEG analysis, is not well provided. Though the waveform analysis can provide effective time domain information, the overall power trend in each epoch or contextual information of the specific waveforms are not well

provided. In pattern recognition analysis, because of the variety in feature vectors, an individual specific event such as transient peak or a specified waveforms cannot be identified properly. It is clear that none of the methods described to this point is good enough to perform the complex procedure of the sleep EEG analysis.

In conclusion, in computerized sleep analysis, individual waveforms such as alpha, beta, delta, sigma, etc. can be detected, but certain waveforms and sleep states, such as REM, slow wave or K-complex can be recognized and interpreted only by considering multiple channel activities and/or the previous and following epoch activities. That is, after detecting an individual wave, a contextual analysis of multichannel EEG/EOG signals is required.

CHAPTER 3
EXPERT SYSTEM APPROACH

The previous chapter discussed the advantages and the difficulties or weaknesses of the conventional methods for sleep EEG data analysis. We notice that the procedure for the sleep EEG data analysis needs two processes-namely the visual perception of the waveforms and the contextual understanding of each waveform.

For visual perception flexible waveform criteria which can be adjusted depending on the subject or required analysis are needed. For instance, the amplitude threshold value for a certain waveform detection can be different depending on the subject. For the contextual understanding of each waveform or contextual analysis of each channel activity, a set of rules is required. These rules form the domain specific knowledge base for EEG signal analysis using all the channel information and subject dependent data. The channel activity for an epoch is described using these rules. The flexibility of the waveform detection and expert rules for contextual understanding of the waveforms are the key elements to be encoded in the present EEG analysis system.

Recently, in the field of Artificial Intelligence, various kinds of expert systems were developed to solve the problems which require the knowledge base of a human expert

and a deduction mechanism for decision making [Ba82a] [Ba82b] [Co81] [Mi81] [Ge83] [St82] [We84] [Mc82]. In the following section Expert Systems are briefly discussed to show their present status and how and to what extent such techniques can be utilized for the sleep EEG data analysis.

Background

Most of the early work in AI research centered around an attempt to find a general (domain independent) problem-solving strategy (GPS) in a belief that a few reasoning mechanisms with powerful computers can give the solutions to domain-independent problems. Though this GPS research demonstrated the important role of the fundamental techniques such as various search methods, knowledge representation methods, and problem decomposition into subgoals, ultimately it led to the view that they could not offer sufficient power to solve most complex problems [Ne69]. Many of the researchers have instead focused on simple or single domain problems which require incorporation of large amounts of domain specific knowledge. These are the Expert Systems.

Expert systems deal with various problems which fall into one of the following categories: data interpretation, prediction, diagnosis, monitoring, planning, design, debugging, repair, instruction, control, etc. [Ha83b]. Expert systems solve domain specific problems at the level of human expertise utilizing accumulated expert knowledge, comprehensive knowledge representation and inference schemes. Expert knowledge provides the key to expert performance, while

comprehensive knowledge representation and inference schemes provide the mechanism for its use [Fe77]. Most of the work in building an expert system involves capturing the expert knowledge and then forming a knowledge structure for the deduction mechanism or inference schemes to use that expert knowledge efficiently.

Various knowledge representation techniques and inference schemes [Wi79] [Ri83] [We84] [Ni80] have been developed for different domain problems. The structure of the inference scheme is usually closely related to the method of knowledge representation. Once the rule language has been defined, the inference engine can be built in terms of the general framework. Rules for the domain-specific knowledge are written in the previously defined language format or grammar. In an ideal expert system, the inference engine and knowledge base are separated for system development. For instance, if a certain domain knowledge is represented using production rules, the knowledge base can be changed without affecting the inference engine. R1 (an expert system for configuring computers) [Mc82] is a good example. Though the inference engine and the knowledge base are written as rules, the knowledge base is changed without affecting the control strategy of the inference engine.

Recently, some of the expert systems and general-purpose representation languages have been developed as expert system design tools. These expert systems evolved from their previous expert systems, and the specific languages

have been used or developed for their own expert systems [Ha82b]. Although each of these expert systems has been generalized to adapt different domain problems, their performances are limited by the previously defined rule language and the control structure embodied in the inference engine. Although the general purpose representation language has fewer limitations, the appropriate rule language and control structure have to be defined in correspondence with appropriate knowledge structure to represent their domain specific knowledge.

Some Expert Systems Applications.

Several expert systems are reviewed in this section to provide an insight into the knowledge representation techniques and inference schemes that can be utilized in building expert systems. These systems (CASNET, MYCIN, PIP, INTERNIST, HEARSAY-II) which were developed in the early age of AI for medical consultations are basically based on the production system and are classification models. Though the following expert systems could not find their way into every day use, they demonstrate that, depending on the problem domain, an appropriate knowledge representation technique and inference scheme should be utilized.

CASNET/Glaucoma Consultation System

A causal associational network [Ku80] was developed to represent knowledge of reasoning about diagnosis, prognosis and treatment selection in opthalmology. Production rules were used to represent the relationships between observation

and diagnosis in a particular type of semantic network. The representation of the domain knowledge is composed of three layers: observation, pathophysiological state, and disease category. Each observation is mapped into the corresponding pathophysiological state through the implication links. The classification links connect each pathophysiological state to a disease with a one to many relation. In pathophysiological states, the causal links are used to relate cause and effect. The causal relations, with associated degrees of strength, express the mechanism of a disease and their modifications under various regimens of treatment [Ku80]. With causal ordering information, the most likely cause for a particular set of patient data can be easily deduced by traversing the causal network.

The reasoning control strategy of CASNET starts in event-driven fashion. Utilizing the incoming clinical data and a thresholding evaluation mechanism, patient-specific interpretation models or hypotheses are formed for confirmed and undetermined states. At these states the causal model is used to constrain the search space for possible hypothesis. The next question choice is "hypothesis-driven". When all the available data are accumulated, the system makes a final decision about diagnostics, prognostics, and treatment categories in a purely deterministic fashion. However, the CASNET/Glaucoma model never became widely used. It was pointed out that the problem domain was related in scope to many additional opthalmological problems which were

not covered fully in the original model. Also, many consultation problems did not easily fit into a causal framework.

EXPERT as an expert system design tool evolved from CASNET. This production rule language maps findings onto hypothesis and solves problems using a simple classification model. In this system the production rules are classified in terms of the three types of logical relationships between findings and hypothesis: FF (finding to finding), FH (finding to hypothesis), and HH (hypothesis to hypothesis) rules. FF rules are used to establish local control over the sequencing of the questions i.e. to specify deterministic logical relations between findings whose truth value is already found. FH rules are used to assign a certainty factor to a hypothesis from the findings. FH and HH rules are used to infer hypothesis from findings and hypothesis respectively. Using these simple logical relations the reasoning rules and expert knowledge are described in production rules. While these simple rule classifications make EXPERT easy to use, it is limited in the application to the different domain problems which have different knowledge structures. However, EXPERT has been used in numerous applications including medicine, oil well-log analysis, laboratory instrumentation, and computer fault diagnosis [We84].

MYCIN/Infectious Disease Treatment Consultant

MYCIN [Sh73] [Sh75] [Sh76] [Sh79] was developed to determine the likely identities of pathogens in patients with infections, and to assist physicians in the selection

of a therapeutic regimen appropriate for opposing the organism under consideration. Production rules with certainty factors were used to represent the control strategy and domain specific knowledge. The certainty factors, which are an approximation of conditional probabilities, are used to model the inexact reasoning process of medical experts. The certainty factor is defined as MB - MD (measure of belief and disbelief). MB and MD are defined in terms of conditional and a priori probabilities [Sh75]. The certainty factor relates possible conditions to associated interpretations with weighted relations ranging from -1 to +1. In the predicate structure (IF-THEN) representation of the knowledge the antecedents and actions are expressed with the attribute-triple pair with certainty factor or attached procedures. The associative triple pair consists of "attribute", "object", and "value". In MYCIN the objects in the triple pairs are the domain entities which are organized into a hierarchical context tree. Each finding or observation is related to the value of the triple pair. The triple pairs represent medical facts and hypotheses or related diseases. This uniformity of knowledge representation made it system possible to separate domain specific knowledge and control strategy in this system.

As with most production systems, a goal-directed backward chaining strategy was used. To find the highest level goal, such as treatments for all the infections of the patient, MYCIN starts by testing the condition of the highest

level rule against available data, or requests data from the physician. Every possible condition is checked for validity and other rules can be inferred recursively at this stage, generating subgoals to be confirmed with findings. MYCIN also utilized certainty factors in the deduction process. It is a hierarchical AND/OR tree with certainty factors, combining uncertain assertions and heuristic cumulative function for decision making [Sh75], using a fuzzy logic function. By looking into a stack of production rules used, MYCIN can answer questions about decision making such as how and why. The premises and the conclusions of the rules are utilized to generate a natural language like answer.

The MYCIN system has been generalized to EMYCIN (Essential MYCIN) [Me81] for domain-independent use. EMYCIN has a rule compiler which takes advantage of knowledge available from static analysis of the rule set to produce a more efficient representation. It constructs a LISP program which eliminates redundant computation of a same premise clause or conditions in different rules. Thus the rule compiler makes the system use an efficient deductive mechanism. This rule compiler works like a rule manager to make rule addition, debugging, and explanation easier. EMYCIN has been used to build various consultation programs. In an application [Br80] of EMYCIN, some problems about backward chaining and parallel searching with a negative certainty factor were reported. With the backward chaining control structure, the grouping of questions is determined by the order of the

appearance of the parameters or variables in rule. That is, data has to be used in a serial rather than parallel fashion. And in parallel search, the certainty factor has to be adjusted to be dependent between subtrees with the same node.

PIP/Present Illness Program

PIP was developed to take the history of the present illness of a patient with edema [Pa76]. The present illness forms a basis upon which a physician builds his diagnosis and bases his treatment schedule. PIP consists of four building blocks: patient specific data (input data), supervisory program (inference engine), short term memory (working memory, hypothesis under testing), and associative memory (knowledge base). The supervisory program is a control program which guides the system in taking the present illness and controls the operation of various subprocesses, such as selecting questions or relevant rule selection to build a coherent model of the case. The short term memory is the site in which the patient's data interact with the knowledge base in associative memory. The associative memory as a system knowledge base is organized into many frames. Each frame is a structure with a name and a number of slots, which can be filled by various properties, logical and semantic relations, and associated inference rules. Frames in the associative memory represent the diseases, clinical states or physiological states. The frames are linked into a complex network. The links of the network

represent the kinds of association between the prototypical findings, such as "caused-by", "must exclude", "component of", etc..

The reasoning starts with the presentation of the patient's complaint, which is characterized by the supervisor utilizing the procedure kept in the knowledge base of long term memory. The characterized patient's complaint activates the corresponding rules. Then a hypothesis is generated, moving frames from the associative memory to the short term memory, where each frame recursively induces related frames in the network. To evaluate each frame or hypothesis a scoring process is performed. In the scoring process, certainty factors contained in the frame are used to reflect the likelihood that various clinical findings exist in the given disorder. This score has two components: a measure of fit, i.e., matched findings to expected findings, and the ratio of the number of findings matched to the total number of findings. In this evaluation process, if necessary, more information is collected to fill each slot of the frame. It has been reported [Sz79] that the generalized scoring function and the threshold value for termination procedure caused difficulties in generating lines of the reasoning process.

INTERNIST-I/Diagnostic Consultant in Internal Medicine

INTERNIST-I [Mi82] has been developed to aid the physician with a patient's workup in order to make multiple and complex diagnosis in internal medicine. The INTERNIST-I

attempts to form an appropriate differential diagnosis in a selected group of observed findings, the differential diagnosis of which forms what is assumed to be a mutually closed set of diagnoses. The search space of INTERNIST-I is very large and highly structured as in general internal medicine. The individual disease is a building block of the knowledge base. For each diagnosis entered into the system, a disease file is generated to include findings that have been reported to occur in association with disease.

INTERNIST-I has knowledge representation of weighted links between findings and diseases. The diseases have hierarchical structure from general to specific. The "evoking-strength" links finding to diseases and "frequency" links diseases to findings. The evoking-strength links describe how strongly the findings explain the diagnosis. The frequency is an estimation of how often patients with the disease have the findings. The other links are causal links which connect diseases to diseases. Each finding is assigned a disease independent "import" weight. The import is the global importance of the findings or manifestation, that is, the extent to which one is compelled to its presence in any patient. Most of the rules are expressed as link relations.

The reasoning strategy of INTERNIST-I begins in data driven fashion. After partitioning general findings into problem areas, a set of selected diseases are tested against the partitioned findings to obtain highly weighted diseases.

Each disease is scored positively and negatively based on the findings found and findings not found respectively. Heuristic principles, such as diagnosis by exclusion, are employed to resolve each of the differential diagnostics. To control the number of induced hypotheses, different strategies are used depending on the number of competing hypotheses. However, the most highly ranked hypothesis is selected using explained findings. Then the program repeats the cycle with unexplained findings until all the findings have been accounted for. Several problems were reported, including the inability to construct differential diagnoses spanning multiple problem areas, the inability to explain its reasoning, inability to reason anatomically or temporarily, and the occasional attribution of findings to improper causes. The successor to this system CADUCES is still being developed [Mi82], (see [We84]).

HEARSAY-II/Speech-Understanding System

HEARSAY-II was developed to understand connected speech [Er80]. Speech understanding problem needs multilevel information transformations from electrical voice signal to recognized sentence. The levels include parameter, segment, syllable, word, word-sequence, phrase, and data base interface levels. At each level different knowledge sources are distributed and utilized to induce higher level knowledge sources or hypotheses. The knowledge sources perform a variety of functions, such as extracting acoustic parameters, classifying acoustic segments into phonetic classes,

recognizing words, parsing phrase, and generating and evaluating predictions for undetected words or syllables. Production rules are used to represent each level of knowledge.

HEARSAY-II attempts to reconstruct a speaker's intention from hypothetical interpretations obtained at various levels of abstraction. The goal is to construct the most credible overall interpretation. The basic cycle of the operations in construction are hypothesis generation, hypothesis combination, and hypothesis evaluation. At each step in the construction, each knowledge source builds a larger interpretation, adding their constraints to the interpretation. These constraints are used to resolve uncertainty in the data and in the knowledge sources themselves. For this incremental problem solving method, the independent knowledge sources need to communicate with each other through a "blackboard" which contains currently competing hypotheses and data of various knowledge sources. From the blackboard each knowledge source gets information about the intermediate state's partial interpretation and generates a hypothesis on the blackboard. The information on the blackboard is also divided into various levels depending on the level of the knowledge sources. Whenever a hypothesis is recorded on the blackboard, a scheduler with data-directed control regime which utilizes opportunistic problem solving method [Ni82] makes a schedule for the next knowledge source activation. To avoid possible combinatorial explosion of

knowledge source invocation the computing resources are allocated to the most important and most promising action. A heuristic schedular [Ha77] calculates a priority for each action and executes the waiting action with the highest priorities. Calculation of priority is based on the blackboard information and the effects of each action of the hypothesis.

Based on the HEARSAY-II architecture, HEARSAY-III [Ha83b] was developed as a domain-independent framework. The HEARSAY-III structure is similar to that of HEARSAY-II, consisting of a relational data base system and its control mechanism. It also utilizes a blackboard structure for system wide knowledge source communication.

We have seen several examples of expert systems and expert system design tools (more such systems are described elsewhere [Ba82a] [Ba82b]). Most of the systems discussed work only in a narrow domain, and their performances are not so successful as to have led to their daily use. In CASNET, for example, the problem domain was not clearly understood in the initial design stage. The problem domain of glaucoma itself is overlapped with the other opthalmological problems in which the different knowledge representation schemes or the different control mechanisms are required. Also the knowledge base about the other opthalmological problems was not fully supported in the original causal network. In MYCIN and PIP there are no general scoring rules which assign a precise certainty factor to a specific finding. The

heuristic rules cannot be always properly represented using certainty factors or probabilities. The effects of a certainty factor, i.e., the propagation of a certainty factor in a decision tree, were also not covered completely. The linear relationship between different nodes in a decision tree is not always linear. Therefore, it is very difficult to change the knowledge basis, i.e., the system refinement which is essentially required in the development process is very difficult. Also, in HEARSAY-II the uncertainty of multilevel knowledge sources causes the same scoring problem and propagation problem of a certainty factor of heuristics. In INTERNIST and HEARSAY-II the problem domain of internal medicine or of the vocabulary to be recognized is too big to handle using specific methods and limited computational power. The large search space has to be divided into subspaces of the high levels of expertise. There is too much knowledge to be encoded in the knowledge base for these systems. The multi-symptoms of complex diseases could not be fully covered in INTERNIST. The limitation of the knowledge base in HEARSAY-II permits only about one thousand spoken words to be recognized with a speaker's trained speech data.

However, each system has a different architecture corresponding to their domain specific problem. That is, the required expert knowledge and the way the human experts use their knowledge to solve problems are different and can be structured appropriately depending on the characteristics of the problem domain. Surely, this is the reason why expert

systems have different architectures and the key point that has to be considered in any expert system design.

Expert Systems Design Considerations

In this section the general problems of the expert system and general design steps are discussed to characterize the design problem of an EEG expert system.

The major problems of expert systems are [Da82]

1) To separate the domain dependent knowledge base from the inference mechanism used for reasoning and controlling the system (separation between knowledge and meta knowledge for knowledge accumulation): In general, the separation of the inference mechanism from a knowledge base is not always clear. The inference mechanism itself as a knowledge base may be embedded into the system knowledge base.

2) To develop logically powerful and expressive representations for describing concepts, facts, and procedures (knowledge representation): The knowledge representation techniques are chosen based on the usage of the particular knowledge in the system. There is no best way for knowledge representation. Depending on the characteristics of the problem domain the proper knowledge representation techniques have to be used for system efficiency.

3) To capture the expert knowledge about specific inferences or decisions in the form of modular rules without confliction or redundancies among the rules (knowledge base management): Modularity of the knowledge is dependent on the structure of the knowledge. Modular rules are not always

possible.

4) To develop a general inference mechanism to control reasoning (deduction mechanism for decision making): The more the framework is generalized, the more difficult it is to implement the domain specific expertise.

5) To develop methods of facilitating user interaction with the system by a way of normal conversation (natural language interface): Natural language processing is another big area in AI. A natural language like user interface makes the system more intelligent.

As previously discussed, the problems of expert systems result in a compromise between system performance and the ease of the computer implementation, or between the system expertise and the size of the problem domain. That is, the more the system is generalized, the less is the expertise.

The process of building expert systems can be described by the following steps:

1) Problem identification: Problem characteristics are determined, such as area, scope, possible problem solving methodologies, required resources, constraints, and specification of goals.

2) Conceptualization: A detailed description of the information flow is determined, such as information transformation in each intermediate stage, data and hypotheses control mechanisms, and possible implementation methodologies.

3) Formalization: The knowledge representation techniques, system rule language, and inference mechanism are selected

appropriately.

4) Implementation: The rules that embody the knowledge are implemented on a computer.

5) System refinement: The system is evaluated using field data and modified according to the rule modifications.

EEG Expert System Design Concepts

In this section the basic concepts of the EEG expert system design are described based on the design steps described in the previous section. A detailed description of the system design is given in the following chapters.

1) Problem Identification: The system will interpret multichannel EEG (three EEG and one EOG channels) for sleep analysis. Epilepsy waveform detection and sleep disorder diagnosis will be included in the future. The knowledge base of the system consists of specified waveform detection techniques, EEG signal understanding rules, and clinical knowledge about the corresponding diagnosis.

2) Conceptualization: The system has three different knowledge levels: the waveform detector outputs of the EEG signals, the channel activity description, and the synthetic report level. The specific waveforms are detected from the EEG signals using digital waveform detectors. The channel activities are described based on the waveform detector outputs. A synthetic report is generated using the channel description list and corresponding clinical knowledge bases.

3) Formalization: At the waveform level each waveform is expressed as a hexadecimal character. All channel activities

at an instant form a word. Using these words the channel activities are described in the channel description level. The channel description list is utilized for synthetic report generation. To realize these processes in the main processor, the language understanding mechanism is utilized. The EEG signal language is defined and the EEG signal compiler is designed using existing software of the UNIX operating system. To display the line of reasoning, the channel description list is saved in a system utility file. On the synthetic report level, interpretive knowledge such as EEG rule understanding is incorporated with the channel description list to generate a synthetic report. The production system and frame scheme are used to represent the EEG signal understanding rules and the clinical knowledge.

4) Implementation: One signal processor is used as a frontend processor to generate "tokens". The frontend processor is implemented using a TI-990/101MA board level microcomputer and an A/D board (RTI-1241, Analog Device). A PDP-11/23+ microcomputer will be used as a main frame. The token generator is written in assembly language after considering the data processing speed. For the token processor the channel description module is written in "C" language. This module is automatically generated using a compiler generator mechanism. The synthetic reporter is written in LISP for processing a channel description list.

5) System Refinements: The system performance is tested using previously recorded EEG data tapes. Once the prototype

is built the system can be tested against human experts. System refinements such as rule modifications can be continued up to the level of a human expert. Various user friendly modules can be added at this stage.

Although these are the expert system design processes for the target system, in the present system not all the features are implemented. In the next two chapters the design details of the EEG expert system are discussed and the system status is described precisely.

CHAPTER 4
DESIGN AND IMPLEMENTATION - I

This chapter describes the human EEGer's sleep data recognition procedures in detail to provide an appropriate EEG expert system model. After discussing the system information flow and structure, each system block design and implementation of the frontend processor are described in detail. The main processor, the system status and future system expansion are discussed in the next chapter.

Design Background

The human EEGer analyzes the sleep EEG data by looking at an epoch (one minute or thirty seconds data) of the data. In epoch, the specified waveforms are recognized by considering all the channel activities. Sleep staging is performed by looking at all the activities in an epoch. If ambiguity occurs within an epoch of data a couple of the previous or following epochs are checked to see the change of waveform activities, such as the amplitude variation or wave rhythm change between consecutive epochs. In the epoch analysis specified waveforms, such as alpha, beta, delta, spindle, etc., are recognized as a group of waveforms by the number of local peaks and their duration. For instance, to detect an alpha waveform (8 to 12 Hz), the number of the waves is counted in a certain interval, which may be a one or two second interval. If at least six consecutive waves,

whose duration lie in the range of the alpha wave are found, then they are regarded as alpha waveforms. Of course, these numbers do not precisely represent the visual analysis criteria. Also, the selected alpha waveform itself cannot be clearly defined. They can be consecutive waveforms or a group of split waveforms [Sm78]. In some aspects, it is subjective whether a certain waveform is regarded as an alpha waveform or not. However, by adding up the alpha waveforms in an epoch the total alpha time of an epoch is obtained. In the adding procedure, because of the ambiguous waveforms the human EEGer cannot make a clear distinction of the waveforms and the exact amount of the total alpha activity time cannot be obtained. Usually an approximation of each second is made and added up to get the total alpha time in an epoch. The sleep staging rules are applied on this total alpha time.

On the other hand, in the case of the delta wave (0.5 to 2.0 Hz), usually the number of peaks in a certain interval are counted or the period between the local peak points is compared to the interval of 0.4 seconds, denoted by lines on the data sheet. In this case the local peaks of the high frequency components or the fast activities are just ignored or smoothed to count only the slow wave components. The total delta activity time is obtained in much the same way as the alpha time.

While the alpha wave and delta wave recognition procedures require the checking of only one channel, the REM

waveform recognition procedure needs to check all the channel activities. When a slow wave appears in the EOG channel, the REM waveform recognition is totally dependent on the other channel activities. The slow wave activities of the frontal channel usually affect the EOG channel waveform. When a K-complex waveform occurs it appears on more than two channels simultaneously. The EOG channel can be one of them. When delta activities appear on the occipital channel, the EOG channel also usually shows slow wave activities. Thus the EOG channel slow wave has to be interpreted by considering all the channel activities. On the other hand in the REM period detection, the epoch which is between REM periods is regarded as a REM period without considering the slow wave existence on EOG channel. Therefore, for sleep data interpretation all the channel activities have to be checked and in addition the preceding and the following epochs have to be considered simultaneously.

Then how can these processes be realized on the computer? As previously shown, the procedures or steps of the human EEGer's sleep EEG data analysis require individual waveform recognition and contextual analysis of these waveforms. Especially for the contextual analysis of the waveform the descriptive knowledge or procedural knowledge has to be utilized with the channel information. To approach these kinds of waveform recognition and interpretation problems, an expert system is designed. Each step of the sleep EEG analysis is simulated using a hybrid system.

A signal processor which consists of an A/D board and a microcomputer detects sleep waveforms. A main processor which is a software program of a minicomputer performs contextual analysis of the detected waveforms. Though this system does not exactly simulate the EEG data processing procedures of the human EEGer, in every step of the recognition procedure an attempt is made to incorporate the corresponding techniques of waveform detection or knowledge base of waveform interpretation to the system design. Each specified waveform is detected using a waveform detector which simulates the detection process of the human EEGer and then the waveforms are interpreted utilizing a contextual analysis which utilizes all the channel information and the subject dependent data. Although there are minor changes in the waveform detection criteria, the waveform detection techniques are mainly based on the techniques of the SAHC [Sm78] [Pr79] which performed well in the sleep EEG waveform analysis.

In the present system all the analog filters and hardware detectors of the SAHC are converted to software programs (assembly language on a TI-990/101MA board level microcomputer). Slow wave detectors are added to each channel for the REM period detection. For the contextual analysis of the sleep EEG signal the signal language is defined and semantic analysis is performed. The computer compiler mechanism is utilized for signal language understanding and the standard software (LEX, YACC) for the compiler

generation is adapted for signal compiler generation. The detected waveform patterns and the relationships between patterns are matched to the tokens and the syntactic structure of the compiler, respectively. The analysis of these syntactic patterns in the detected waveforms corresponds to the semantic action of the compiler. Furthermore, the automated procedure of compiler generation gives great flexibility in rule management. The waveform interpretation rules in signal grammar and semantic action file can be changed independently of the other part of the system. Automatic system generation is attempted to simplify the frequent rule changes which essentially occur in the expert system development processes.

Although the target system is to be implemented on a PDP-11/23+ microcomputer with UNIX operating system, the current system is developed on a GOULD Concept 32/8780 minicomputer using the UNIX operating system. The portability of the UNIX operating system and the use of the standard software and language "C" make system conversion a simple process. For the data pipeline in the system the system shell command is developed using the UNIX system shell command. The language "LISP" is selected for the processing of the channel description list and the system knowledge base for the sleep disorder diagnosis. LISP will enhance the capability of the interactive user interface and system reasoning.

System Information Flow

Fig. 4.1 shows the system information flow. The key concept of designing the system is to simulate the interpretation procedures of the electroencephalographer in analyzing sleep data. From the multichannel EEG/EOG inputs, a synthetic report of the sleep data analysis is generated.

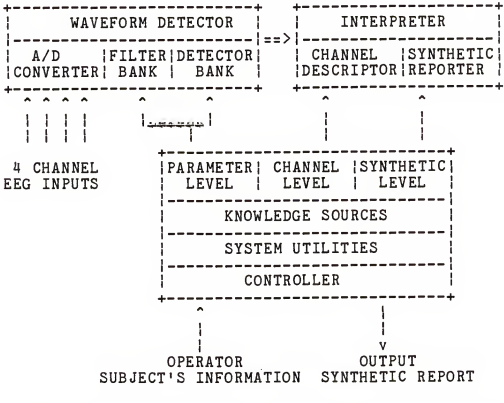


Fig. 4.1 Block Diagram of System Information Flow.

The system is characterized as a distributed multilevel (or cooperative) knowledge base expert system. In each step of the data understanding or the interpretation procedure the various knowledge sources interact with each action module, such as the filter and detector bank, channel

descriptor and synthetic reporter. Each action module accepts the data from the previous module and processes the data utilizing its knowledge source. In this process the data flow occurs only in one direction between the modules, from low level to high level.

The knowledge base is distributed throughout the system at each information processing level such as parameter level, channel level and synthetic reporter level. In the parameter level the information about the data sampling such as channel assignment, sampling rate and channel multiplexing is incorporated in a data handling module (interrupt handler and main program of the token processor of the present system, refer to Appendix 2). In the filter bank (Fig. 4.2) the required cascaded filter information such as amplitude characteristics, phase response, and transient response are incorporated in each filter module. All the detector criteria for the specified waveforms such as the required window size of the specified waveform detection, wave periods, number of consecutive waves, amplitude threshold value, hysteresis, minimum display time, wave period averaging process, etc. are incorporated in each detector module. In the channel level the channel description rules for the specified waveforms, such as waveform significance resolution, waveform relation resolution in time domain and waveform smoothing for total activity time calculation are incorporated in the channel descriptor. The channel description rules are in the rule files (lex source files,

grammar files and semantic action files of the present system) of the channel descriptor module. In the synthetic level an epoch description rules such as REM period recognition and sleep staging rules are incorporated in the synthetic reporter module.

The system consists of two functional blocks: the frontend processor and the main processor. The frontend processor (which is also called a token generator), as a signal processor, includes an A/D converter, a filter and a detector bank. The main processor (which is also called a token processor), as an interpreter, includes an interpreter, knowledge sources, system utilities, and controller. The frontend processor detects the specific waveforms and generates the token stream. In the detector bank all the waveform detector criteria are embedded into the different detector modules. The main processor interprets the token stream and generates a written report. The knowledge base of the main processor consists of channel description rules and REM detection rules for sleep staging. The knowledge base will be expanded to include the sleep disorder diagnosing rules.

The signal flow of the present system is as follows. The analog input signals from the EEG/EOG amplifiers are digitized by an A/D converter and passed through the digital filter section. The cascaded filters (Fig. 4.2) with different sampling rates [Pr79] approximate the first step of waveform recognition of the human EEGer by selecting the

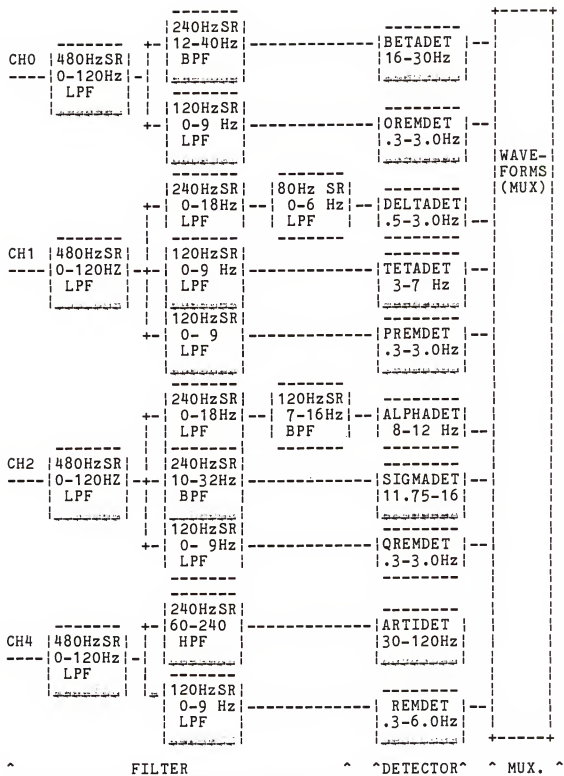


Fig. 4.2 The Frontend Structure of the System.

specified frequency component from the EEG/EOG signals. The corresponding detectors then detect the specified waveforms appearing in the corresponding channel utilizing waveform criteria [Sm78]. All the detector outputs are sampled simultaneously and coded into a two byte word. This two byte word forms the input token to the channel descriptor (Fig. 4.3). This process in which all the channel information of the same instant are multiplexed, provides the way of information checking across the channels at an instant. These tokens are then interpreted by the interpreter of the main processor using heuristic rules derived to mimic the human expert.

The interpreter consists of a channel descriptor and a synthetic reporter. Each of these modules utilizes the corresponding knowledge sources to process the token stream. In the channel descriptor each channel activity is described, based on the total activity time of the specified waveform in each channel. The token stream which includes all the detected waveform information is regarded as a signal sentence and interpreted to generate a synthetic report in each epoch. The descriptive words for the channel activities are previously selected depending on the waveform characteristics and the required analysis.

Fig. 4.3 shows the interpreter structure of the main processor in the system. In the channel decoder the lexical analysis of the proper tokens is performed based on the waveform or channel priority, namely selecting a correct

detector output. The channel priority and waveform priority are resolved in the decoder. The syntactic and semantic analysis of output relationships between different tokens of the same channel is performed on the output of the decoder.

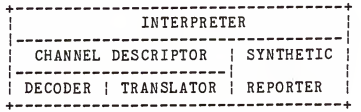


Fig. 4.3 Block Diagram of Main Processor
(Token Processor)

Then the translator generates the description list of all the waveform activities of the corresponding channel. Heuristic rules for interpreting these relations are applied to this stage of analysis. The channel description list is used for the synthetic reporter.

The synthetic reporter makes the final synthetic report after considering all the previous channel descriptions. Many of the technical rules for interpreting the EEG data are implemented in the synthetic reporter based on the hybrid production system and frame scheme [Wi79]. The rules for the synthetic channel description are implemented using a production system [Ni80]. The structure of the channel description rules in the production system simplifies the connection between the EEG data and the other knowledge base, such as sleep disorder diagnosing rules. Both the

channel description rules and sleep disorder diagnosing rules will be put into a rule file. This will unify the system rule management. The frame scheme [Mi76] is considered for the sleep disorder diagnosing rules of EEG diagnosis (not used in the present system, where only the basic inference mechanism is implemented). A clear distinction between the knowledge base for the channel descriptor and the knowledge base for the synthetic reporter cannot be made though they have different mechanical structures for information processing. Some of the rules can be implemented in either place without affecting the final decision. The detailed discussion of each block design and implementation are given in the following section.

Frontend Processor Implementation

The frontend processor generates tokens every quarter second utilizing an A/D converter, a filter bank, a detector bank, and a multiplexor. The frontend processor except for the A/D converter is implemented on TI-990/101MA board level microcomputer. The architecture of the independent work space pointer of the TI-990/101MA microcomputer causes independent operation of each filter and detector and speeds up the linking operation of each module. The expansion memory board of 32K bytes memory is used for the program space (4K bytes) and the generated token storage space (28K bytes). The on-board memory of 4k bytes is also used for the program space. The functional flowchart of the frontend processor is shown in Fig. 4.4. The program consists of a

main program, 12 filter modules, 10 detectors, and an interrupt handler.

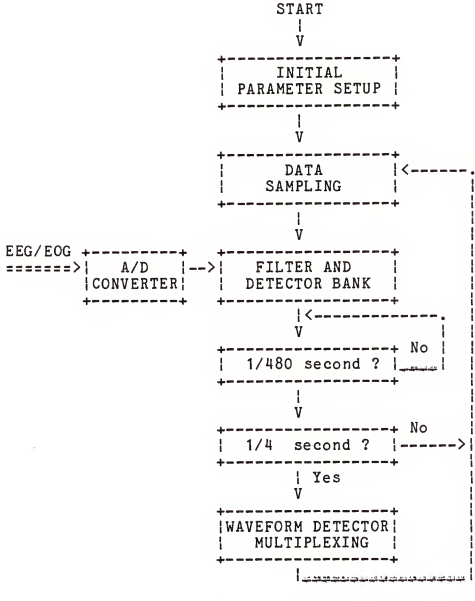


Fig. 4.4 Functional Flowchart of the Frontend.

In the beginning of the main program parameter initialization is performed. After setting up the parameters, each filter, detector, and interrupt handler are called in the

main program according to the previously selected order. Once the program starts, the third level interrupt is generated every 1/480 second. In that interval the related filters and detectors detect the specified waveforms. Then the multiplexor generates tokens every quarter second, using detected waveforms. Each functional block is discussed in detail in the following sections.

Initial Setup

In the initial setup procedure various addresses are defined, such as that of each filter and detector module, I/O port of each module, interrupt handler, A/D converter base address, starting address of the processed data storage, and various clocks and counters. All the variables are also initialized, such as sampling rate, control parameters of the A/D converter, and various clocks and counters (see main program in Appendix 1). To control the A/D converter, first the CRU (communication register unit) base address is set to A/D converter base address and various control registers of the A/D board are set. The CRU of the CPU board is used to control the mapped I/O of the system. Using a register in the main program the auto scanning mode of the multiplexor is disabled to check only four input channels. The sampling port address is handled in the interrupt handler by one of the address pointers in the interrupt handler. The timer is initialized to generate interrupt at 480Hz. The gain of the sampler is set to one in this step. For the proper operation of the interrupt handler the jump

address of the third level interrupt is set to a prefixed address of the system. The starting address of the processed data is loaded by setting one of the interrupt handler registers. That register is used as an address pointer in autoincrement address mode. Most of the registers in the main program are used for the subroutine pointers, making the jump to the subroutine in the main program a quick one. After setting up all the variables the interrupt masks of the PSI (programmable system interface) and the frontend CPU are enabled. The sampling operation starts after getting a third level interrupt signal from the timer on the A/D board. Then A/D conversion is initiated by writing a dummy number on the conversion start register of the A/D converter board. Once the sampling procedure starts then it repeats sampling operation at 480 Hz.

A/D Converter

The signals from three EEG channels and one EOG channel are converted into digital signals using an A/D converter (RTI-1241 Analog Device). A 12 bit A/D converter was selected to provide an adequate signal to noise ratio [PRI79]. The base address of the A/D board is selected using jumpers present on the board. The A/D board operation is controlled by the initial setup procedure and by the interrupt handler. In the initial setup procedure, all the parameters are initialized for the A/D board to operate in the multiplexed normal sampling mode after setting up the CRU of the CPU board. For the four channel multiplex

sampling the auto scanning of the input channel is disabled. The input channel addresses are written in the multiplexor address register every four channel sampling cycle of the interrupt handler. A register of the interrupt handler is used for the data input channel control. This register is used as an address pointer in the autoincrement address mode. The polled status mode of the normal operation modes which checks the end of conversion bit to determine the completion of the A/D conversion, is used to maximize the flexibility and the frontend CPU efficiency. In the normal mode operation, once the multiplexor address and the gain are established the conversion is proceeded by writing a conversion command. Then the frontend CPU is freed to perform other tasks.

The A/D board starts the operation with the interrupt generated by the timer. The third level interrupt is generated every sampling interval by the external timer of the TMS-9901 programmable system interface of the A/D board. Then the interrupt handler resets the external timer and makes the A/D converter collect samples at every channel, and then transfers the control over the filter module.

Interrupt Handler

The interrupt handler as an interrupt service routine, is called every sampling interval. First, the interrupt handler controls the data sampling and stores the generated tokens in the memory. Four input data channels are sampled in a prefixed order by keeping the input channel addresses

and are sent to the corresponding filter input. The first filter bank consisting of four 120 Hz low pass filters in each channel, is implemented in the interrupt handler to reduce the subroutine switching time of the cascaded low pass filters. These low pass filter outputs are supplied to each waveform filter with different sampling frequency. In the interrupt handler several timers, clocks, and address pointers are kept for token generation and acquisition. The detector outputs are multiplexed into a two byte word token every quarter second and stored in the extended memory boards. The control signal for the data display on paper is generated every two seconds with a half second turn-on time. The statistics of the processed data, such as total alpha time, beta time, number of spindles, etc. are also stored in memory every minute, After servicing all these jobs, the interrupt handler resets the interrupt mask which enables the third level interrupt of the CPU and the programmable system interface.

Filter Bank

The filter bank consist of 12 filters (except four low pass filters in the interrupt handler). Alpha and delta filters consist of three different low pass and band pass filters in cascade (see Fig. 4.2). The others consists of two filters in cascade. The filtering process is necessary in order for the zero crossing detector to identify a high frequency component superimposed on slow waves, or to eliminate the effects on the zero crossings of high frequency

components superimposed on a slow wave. Actually, in visual analysis of the EEG data the human EEGer also detects a specified waveform having these kind of filtering operations in mind. For instance, the number of local peaks on a slow wave is counted to select a high frequency component or the local peaks of the high frequency component on slow wave are just ignored to select the slow wave.

The filter corner frequencies in each channel are selected depending on the periods of the waves to be selected. Linear phase finite impulse response (FIR) filters were designed using algorithms which eliminate multiplications in the filter implementation by selecting filter coefficients that are the sum of 2**(-n) [Ly77] and using only zeros in the filter transfer function to obtain the specified frequency characteristic of the digital filter [Pr85]. The out of band frequency gain is suppressed by putting zeros closely on the unit circle of the Z plane. This algorithm made the real time operation of the filter bank possible. The digital filters are realized using the work space registers (up to 12 registers). The registers are used as delay units with fixed periods. The delay time corresponds to each filter's sampling rate. The coefficient of the filter is realized by ADD and SHIFT instruction. For instance, 1.25 * A is obtained by the operation of (A + (SHIFTRIGHT (SHIFTRIGHT A))). Filter phase linearity, which can minimize the distortion of zero crossing, is obtained by positioning the zeros at the specified positions on the unit

circle of the Z-plane. In case of narrow band low frequency filters, the undersampling effect is minimized by utilizing a maximum of three cascaded filters, which have different sampling rates. The first filter of the cascaded filters has a higher sampling rate and a wider bandwidth than the others. It makes the first folding frequency range wide. The second filter has a bandwidth which is just larger than the half of the first one (if there is a second filter in the cascaded filter). Then the folding frequency bandwidth can be larger than the case of one filter, even with a low sampling frequency. Also, using this structure the low frequency filters with narrow bandwidth are designed with a smaller number of transmission zeros. This means simpler structures and faster filter operation for the real time data processing. Usually, the narrow pass band filter implementation with high sampling frequency requires many zeros on the unit circle to suppress the gain of out band frequency.

Though the filters are satisfactory for practical applications, the complete elimination of frequency folding from high frequency to low frequency due to aliasing cannot be achieved (see the alpha filter frequency response in Appendix 1). This is mainly due to the maximum possible sampling frequency, which is determined by the speed of frontend processor. Although, using this method the linear phase shift can be obtained, the linear phase intercept distortion [Pe77] is not always achieved. The DINAP [Ba78]

program was utilized to check the frequency response and phase characteristic of each filter. (See Appendix 2 for the filter characteristics and implementation).

Detector Bank

The detector algorithm (Fig. 4.5) used for this system is based on an analog EEG waveform detector which performs well in sleep analysis [Sm75]. There was a minor change in step (5) (see Table 4.1). Step (6) is added for the token generation which samples detector outputs every quarter second. Once a certain waveform is detected then the detector output should be displayed to be sampled correctly. Six detectors for alpha, beta, delta, sigma, theta and artifact are assigned (as shown in Fig. 4.2).

(1) Check zero crossing
(2) If necessary, check minimum amplitude of wave
(3) Check the period of a half wave for slow waves and a full wave for the others
(4) If necessary, use the wave averaging method for the higher resolution
(5) Apply the wave criteria to each wave
(6) Display the detector output for each minimum period

Fig. 4.5 Waveform Detector Algorithm

A slow wave detector in each EEG channel plus a slow wave detector with different bandwidth in the EOG channel are used for REM detection. The waveform detector first checks the zero crossing of each waveform. The sign of the sampled data value is checked for the zero crossing detector.

Table 4.1

Waveform	Sr	A	Period	Wmax	Win	Wout	Average	Dmin
alpha	160	2.4	4-9	6	6	3	(56-84)/6	56
beta	240	2.4	7-160	7	6	3	(50-99)/6	61
delta	60	12.8	14-85	1	1	0	* *	Pr
sigma	240	5.0	14-20	6	6	3	*	Pr
theta	120	3.2	16-32	6	6	3	(85-232)	86
artifact	240	2.8	1-7	6	6	3	*	*
rslow	240	16.8	10-208	*	*	*	*	Pr
oslow	240	16.8	20-176	*	*	*	*	Pr
pslow	240	14.4	20-76	*	*	*	*	Pr
qslow	240	14.4	20-176	*	*	*	*	Pr

Waveform: Sleep waveforms and slow waveforms
Sr : Sampling rate (Hz)
A : Detector amplitude threshold value (uV)
Period : Wave period (number of sampling intervals)
Wmax : Maximum number of waves kept in the detector window
Win : Number of waves for detector on
Wout : Number of waves for detector off
Dmin : Minimum detector display time (number of sampling interval)
Average : Average wave period (number of sampling interval)
Pr : Previous wave period
* : No criteria

Then the maximum amplitude of each waveform is tested. If

the amplitude is too small, the wave is ignored. After checking the amplitude, the period criteria are applied for zero crossing detection. For low frequency waveform detection, i.e., delta (0.5-2.0 HZ) or slow wave(0.5 - 6.0 HZ), half cycle detection is used instead of full cycle detection. The specified waveforms are detected based on the number of wave criteria [Sm78]. For zero crossing detection, the minimum sampling rate fs for a frequency resolution fr [Sm79] [Sm80] is given by

fr=fi*fi/fs, where fi is the input frequency.

Although this condition cannot be satisfied because of the limit on the maximum sampling rates, an averaging method is utilized to increase the frequency resolution of the high frequency waveforms. In the averaging method the specified number of the consecutive zero crossing intervals are summed together. If the consecutive waveforms satisfy the wave criteria then the total zero crossing intervals are averaged and compared to the standard wave period of the corresponding waveform.

For REM detection, three slow wave detectors in three different EEG channels check the existence of slow waves in each channel. If a slow wave exists in any EEG channel, the REM like slow wave of the EOG channel is reexamined in a subsequent context analysis. If a delta wave or a slow wave exists in the occipital or central channel then the slow wave in EOG channel is not regarded as a REM. Also, if a delta wave exists then a slow wave in the previous and the

following 2 seconds is not regarded as a REM. Once the specified waveform is detected, the detector output is turned on for a minimum period of each waveform, providing for the token generation. Because the token is generated every quarter second, the waveform detector output should be longer than this period to be sampled correctly. This minimum display time differs, depending on the waveforms. For instance, the delta detector displays the previous delta wave period and the spindle detector displays 7/16 second (see each detector program in Appendix 1).

Detector Sampling

The waveform detector outputs are sampled every quarter second and stored in a two byte word. Two bytes are used for each sample. Ten bits are used for the detector output, one bit is used for time marking, and five bits are available for future expansion. The relative bit positions of each waveform detector output in a two byte word are represented as follows

[, , ,time mark] [alpha, beta, delta, sigma]

[theta, artifact, ,] [rslow, oslow, pslow, qslow].

Each bracket denotes a hexadecimal number of a four bit binary notation. For instance, the alpha waveform detector output is denoted in the first bit of the second nibble. The simultaneous sampling of the detector outputs makes it possible to analyze waveform relationships between the different channels or of the same channel. The waveform detector outputs form the input token to the main processor. With

this sampling scheme the waveform priority is given easily in a subsequent main processor context analysis. This sampled output of a two byte word is converted to four hexadecimal ASCII characters for data processing of the main processor.

In the next chapter the main processor design is discussed and the system status is described with an example of the data format of each process.

CHAPTER 5
DESIGN AND IMPLEMENTATION - II

The main processor design and its implementation are described in this chapter. Not all of the main processor features described in this chapter are implemented in the present system. The basic frame which demonstrates how the expert system approaches the sleep EEG analysis has been implemented. REM period detection is shown as an example. The system design concepts describe how the system is structured and how the system works. The sleep data analysis example and the current system status are discussed in detail at the end of this chapter.

Main Processor Design and Implementation

The main processor reads the token stream and generates interpreted results every minute. In the main processor the token stream is regarded as a signal sentence which expresses the contained information in a signal language. The signal sentence is interpreted using a language compiler mechanism such as lexical analysis, syntactic analysis, and semantic analysis. The interpreted results will be incorporated with the subject dependent data or knowledge source about sleep diagnosis to generate a synthetic report.

In the present system only the interpreted result of the sleep EEG is utilized in the synthetic report generation. Though different mechanisms can be utilized for the

syntactic pattern recognition, a computer compiler mechanism is utilized for system implementation since the automated procedure of compiler generation provides several useful features.

For instance, the standard software for the compiler generation is used as an expert system development tool of the present sleep EEG expert system. The separated file structure of the compiler generation makes rule management independent in the sleep EEG expert system. This gives great flexibility to rule implementation and makes the inference engine independent of the sleep EEG signal understanding rules. In addition, the automated procedure of compiler generation makes system development rapid. Whenever rule adjustments occur in the system refinement stage of expert system development, the whole system can be automatically regenerated.

For the present system development, a signal compiler is designed for the channel descriptor and a shell program is written for the data pipeline of the channel descriptor. The lexical part of a compiler is used to select a proper waveform token. The syntactic and semantic part of a compiler are used to implement the sleep EEG signal understanding rules. Although the computer compiler mechanism is not sophisticated enough for natural language processing, a limited number of specified EEG waveforms make a high level language compiler mechanism useful for EEG signal language understanding.

The information flow of the main processor is shown in Fig. 5.1. The main processor takes inputs from the frontend processor and the operator.

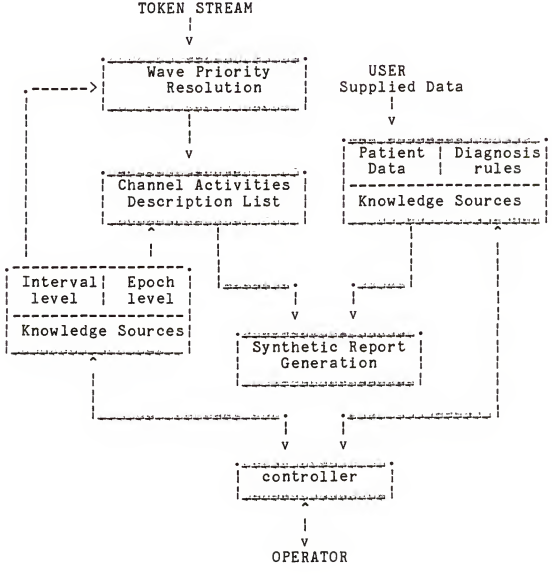


Fig. 5.1 Information Flow of Main Processor

The necessary information for setting the parameters of the interval and epoch level knowledge sources can be obtained

from the operator at the beginning of the session. The parameters set previously cannot be changed interactively in the present system. Some of the rule parameters of the channel descriptor can be adjusted by the knowledge source procedures utilizing this information. After the channel description rules are adjusted, the channel description list is generated from the output token stream of the frontend processor. Then the channel description list is used for synthetic report generation. In synthetic report generation, if the patient data or diagnosing information is required, the corresponding knowledge sources are activated and demand more information from the operator. The information is used when the decision has to be made due to a lack of sufficient data. Only the channel description list is utilized for synthetic report generation in the present system.

The block diagram of the main processor is shown in Fig. 5.2. The main processor consists of a channel descriptor, a synthetic reporter, and a knowledge source about diagnosis (actual diagnostic rules are not implemented in the present system). The channel descriptor describes channel activities of each channel. The synthetic reporter generates a synthetic report utilizing the channel description list and the knowledge source about diagnosis. The design and implementation details of each block are explained in the following block description and implementation.

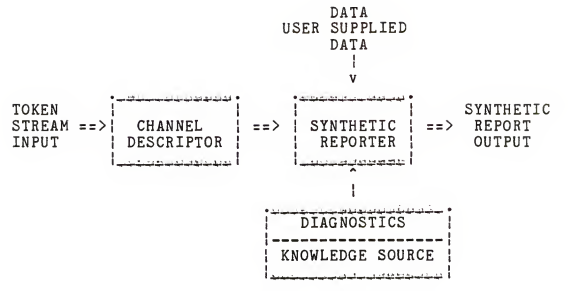


Fig. 5.2 The Block Diagram of the Main Processor

Channel Descriptor

The channel descriptor describes each channel activity in terms of the specified EEG waveforms. For instance, if the total activity time of the alpha wave in an epoch is larger than 30 seconds a channel description list "alpha activity in channel 2 is high" is generated. Fig. 5.3 shows the functional structure of the channel descriptor, consisting of a decoder and several translators. Input tokens are decoded into a set of character streams in the decoder depending on the corresponding detector outputs of each channel. Every waveform detector output is reexamined in each decoder to see if there is a false detection due to other channel activities and if so, it is reconstructed to denote correct waveform information. Each channel activity is described by the translator using the decoded output of each waveform such as alpha, beta, delta, etc.. Two levels

of rules are applied in this process. The first level rules are the first priority rules which are applied independently to the token itself in the decoder. The second level rules describe the relationships between different tokens.

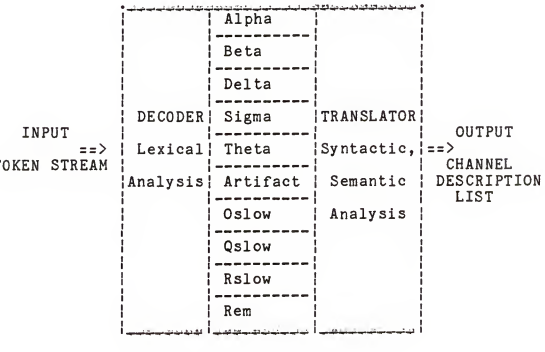


Fig. 5.3 Functional Structure of Channel Descriptor.

In the first pass of the channel descriptor first level rules are applied to all tokens simultaneously. All the channels are tested together by simultaneously looking at each detector output horizontally. That is, the relationships between different channels or different detector

outputs of the same channel are examined. For example, if artifact activity exists in the EOG channel, all the data of the same interval are not reliable. In that case, depending on the previous data, the channel information has to be reexamined. If an artifact is detected then the delta wave detector output of the same interval is ignored in the first pass analysis. Also, if there is a slow wave in the occipital or parietal channel during the same interval, the slow wave output of the EOG channel is ignored. If a spindle waveform is detected with the artifact, it is also ignored and so on. The first pass analysis outputs are the consecutive token stream outputs on which the first level rules already have been applied to resolve the priority between the channels or the detectors.

In the second pass, the second level rules are applied to resolve the vertical relationships of each channel, i.e., the output relationships between different tokens. The relationships between the ouputs of the same detector or different detectors are examined. To check these relations certain rules get first priority. For instance, to decide whether the detected slow waveform output of the EOG channel is truly a REM of the sleep stage 1 or not, the delta wave detector output has to be examined first. If there is a delta wave in the previous or the following two seconds then the detector output of the EOG channel is ignored. The total activity time or the number of each specified waveform is also obtained in this process. For the total activity

time calculation a smoothing process is performed on the detector output. For example, in visual analysis if a discontinuous interval is detected in the alpha waveform detector output, depending on the duration of discontinuity it may be regarded as a continuous waveform in visual total activity time calculation. Three quarter second smoothing is used for slow wave (rslow, oslow, qslow, pslow) and one second smoothing is used for other waveforms in the present system.

The selected word representing a specified waveform activity in the translator depends on the total activity time of the waveform in an epoch. Each waveform activity time or the number of the occurrence for the corresponding description word is given in Fig. 5.4.

	high	medium	low	none
alpha,	30	12	2.5	
beta,	30	12	2.5	
delta,	30	12	2.5	
sigma,*	2	**	1	
theta,	30	12	2.5	
artifact,	20	10	2.5	
rslow	4	2	.25	
oslow,	30	12	2.5	
pslow,	30	12	2.5	
qslow,	30	12	2.5	

* : number of occurrence, others are seconds
**: "medium" is not defined for sigma

Fig. 5.4 Activity Time (number) vs Descriptive Word

To describe the alpha wave activity, the total alpha time

per one minute interval is calculated. The words "high" refer to thoses case longer than 30 seconds, "medium" to cases between 12 seconds and 30 seconds and "low" for cases less than 12 seconds. For spindle activity, the number of occurrences is counted rather than the total activity time. The activity time or the number of occurrences of a certain waveform, which are used for decision criteria, may be adjusted after considering the applied rules for the particular detector.

Channel Descriptor Implementation

As shown in the previous section lexical analysis, syntactic and semantic analysis are required for the EEG signal language understanding, that is for the channel descriptor. In this section how the channel descriptor is automatically generated and how the EEG signal understanding rules are related with the various program modules for channel descriptor generation are discussed.

Fig. 5.5 shows a block diagram of each program module, and the input/output relationship for automatic program generation for the channel descriptor. "Lex" and "Yacc" are standard software library packages on the UNIX operating system which supports the automatic design of language compilers. "Main" is a hand written program in "C" language for calling "yyparse" which is a parsing function. "Lex" [LES75] is the program which generates a lexical analyzer whose control is directed by instances of regular expressions in the input stream. To generate a lexical analyzer using the Lex

program, lexical rules are provided as a lex source. The lex source (lexical rules) consists of two parts: a set of regular expressions and the corresponding program fragments which derive lex action. A set of lexical rules is translated to a program which reads an input stream, coping it to an output stream and partioning the input stream which matches the given expression. As each such string is recognized the corresponding program fragment, such as lex actions, are executed. The recognition of the expression is performed by a deterministic finite automation generated by 'Lex'. In the present system 'Lex' generates a decoder program which reads the input token stream and converts it into an output token stream (yylex in Fig. 5.5) for which waveform priority rules are already resolved.

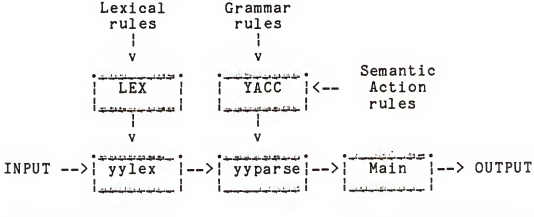


Fig 5.5 Block Diagram for Automatic Program Generation for Channel Descriptor.

In Lex source all the specific waveform tokens are defined in the regular expression and the waveform priority resolution rules are implemented in lex action program segments. More than 60 nonterminal tokens are defined in a lex source; namely 10 waveform tokens, 40 corresponding wave pattern tokens and others. Four rules (seventeen cases) are implemented in the Lex action program fragments of the first pass. More than 40 rules are implemented in the second pass (see Appendix 3 for all source input files, grammar files, parser and semantic action files).

"Yacc" [Jo75] is a program which takes the language grammar and corresponding semantic actions as input source and generates a parsing tree. Yacc reads yylex as input from Lex and takes semantic action when specific rules are recognized. The rules are specified in the grammar in which the input structure is described by the user. The actions are described in the semantic action file.

The input source programs in which the channel activity is described are the key elements in the translator of the channel descriptor. In the present system context free grammar [Ah79] [Ho79] is used to read various EEG signal waveform inputs. Different grammars can be used depending on the signal characteristics of the waveforms to be selected. The channel description rules are implemented in a semantic action file. To describe each waveform activity several counters are kept in each semantic action file. One of the counters keeps track of the continuous off time of a

waveform detector and is then used for a waveform smoothing. Another counter is used to determine the total activity time of an epoch that describes waveform activity. Whenever the specified input token is recognized the counter is increased by one. A certain waveform counter is used to check different waveform activity which is related to its own waveform. For instance, in the rslow semantic action file three counters are used to check the other channel's slow activities. They check the delta activity in the previous and the following two second intervals or slow wave activity in the other channels to see whether a slow wave on the EOG channel is truly a REM wave or not.

In the translator implementation of the present system the "Slrgen" [Lo84] and "Parser" programs are used instead of Yacc. Yacc consists of a parsing table generator and a parser which handles parsing actions such as shift, reduce, accept, and error action. Slrgen is a simple left right parsing table generator which does not accept ambiguous grammar. Parser is a hand written program to handle the parsing table in "C" language. The separated program Slrgen and Parser give more flexibility to handle the various rule implementations. However, to generalize the system generation in UNIX operating system Yacc has to be used.

The actual pass of the each waveform descriptor is shown in Fig 5.6. Each channel descriptor has a common pass of Lex and its individual pass of Lex, Slrgen, and Parser. Although Lex, Slrgen, and Parser in the second pass could be

combined into a single Lex, Slrgen and Parser pass, this structure is selected to give great flexibilities in independent rule management for each waveform description. The corresponding channel description rules are evoked depending on the waveform in each channel. For automatic program generation, a shell command program of UNIX [Bo78] [Bo83] operating system is written in a Makefile (see Appendix 3). All the file management procedures are shown in this file.

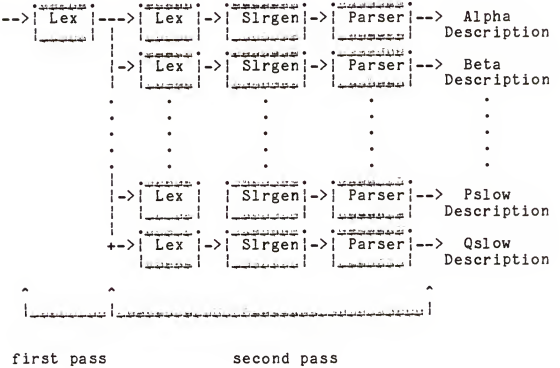


Fig. 5.6 Program Modules used for Channel Descriptor.

Some of the rules are coded into a grammar of the signal specification language and some of the rules are coded

into the semantic action part of the language compiler. For instance, alpha period discrimination is encoded in the grammar file and the total alpha time smoothing, alpha time calculation and the description, are encoded in the semantic action file. In the present system most of the channel description rules are implemented in semantic action file.

Although it is not clear where the rules have to be coded into and although the rule implementation procedures are complex, due to the frequent rule changes in expert system development, standard software provides an excellent tool for designing the present expert system. Specifically, when the waveform in the signal data is well defined it can be considered a problem of understanding the simple signal language. The prominent signal shape to the human EEGer can be regarded as a word in a signal language. The interrelationships of each waveform on various channels can provide a proper grammar and semantic action for interpreting the EEG signal. All the files for generating the channel descriptor are given in Appendix 3.

Synthetic Reporter

The synthetic reporter generates a synthetic report based on the EEG signal analysis results (which is now a channel description list), subject dependent data, and a knowledge source about diagnostic rules (which are not implemented yet). The synthetic reporter will be the heart of the system. The channel description list is one of the knowledge sources in the synthetic reporter. The present

system only contains the REM period detection knowledge.

The structure of the synthetic reporter has to be dependent on the knowledge source to be used because the knowledge structure will affect the structure of inference engine and the reasoning mechanism of the system. Although the structure of the sleep disorder diagnosing rules are not thoroughly studied and cannot be clearly defined, the frame design of the synthetic reporter is described to see how the channel description list can be used in the synthetic reporter.

Fig. 5.7 shows the decision tree of a synthetic reporter.

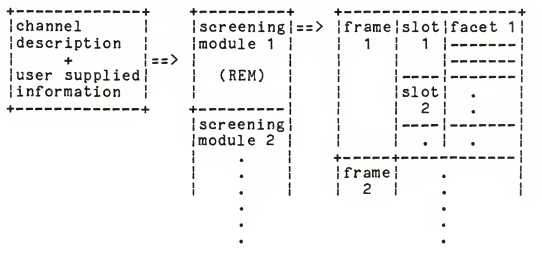


Fig. 5.7 Decision Tree of Synthetic Reporter.

User supplied information forms the input to the screening modules. A screening module is first selected depending on the analysis required. The REM screening module is the first one in the present system. After a screening module is selected, a set of production rules in the screening module gives a best selection of a frame.

For example, depending on the required analysis screening module 1 (REM) can select a proper set of rules to interpret the channel description list (i.e rules for the alcoholics or normal subjects). In this level selection, the branching ambiguity can be minimized by proper grouping of the goal states without overlapping. Once a frame is selected, all the information is utilized to select a final goal state. A synthetic report is then generated based on the information of the final goal state.

For the reasoning process both of the directions of the reasoning need to be used depending on the level of the tree. In the lower level of the decision tree, a branching decision is made using some of the key information of a channel description list and user supplied information. In the upper level of the tree, i.e., the subtree which has a group of final goal states, a model driven or reasoning backward strategy is used to score every member of the group such as a clinical name of a sleep disorder. A confidence factor also can be given to every property in the property list, i.e., a certain symptom of sleep disorder. If there is more than one member which has a score over threshold

value, then all the selected members will be displayed in decreasing order. The selection of the confidence factor of the property and the threshold value for decision making are based on both the human expert knowledge and the other system parameters. This structure was selected to aid future system expansion which will include the analysis of sleep disorders. The synthetic module is written in 'LISP' language to handle symbolic data and natural language like user interface.

Knowledge Representation

Generally, problems in different domains need different knowledge base structures and knowledge base management schemes. Though there exists no general theory of knowledge representation, depending on the application and the kind of knowledge, a good combination of declarative structure and interpretive procedure [Ri83] can be utilized for proper acquisition and retrieval of a knowledge base.

In the present system, the procedural representation method is used to handle the channel descriptor. A hybrid system of a frame scheme and a production system is selected to process the channel description list and the diagnosing rules about sleep disorder. A set of production rules is used in screening modules as branching rules. For instance, for a frame selection, "If REM analysis is required for the normal subject then select frame 1" is used. General signal characteristics of each channel plus some of the user- supplied information constitute the condition part of the

production rules. A screening module contains a certain number of frames. A primitive semantic network in the frame structure is used to represent each group of analysis or diseases. A frame can have different level structure depending on the classification level. A frame contains a group of final states which share some common properties. Each final state has a property list which describes a goal state. Basically, a generalized property list is implemented in a frame structure. Each symptom of a sleep disorder can be described in the property list. If there is a slot of unknown property, information can be obtained from the users in hypothesis assurance session.

The present system has no distinct characteristics of the knowledge representation methods. Depending on the knowledge base which will be included in the future and the reasoning mechanism, the knowledge representation techniques need to be refined.

Explanation of the Reasoning

The explanation of the reasoning is one of the prominent features in the expert system. Although a good reasoning mechanism usually slows down the system, it is necessary for communication with the user. There has to be a tradeoff between good explanation capability and the system speed. The explanation mechanism is heavily dependent upon the system knowledge structure and the language used for building the system.

In the present system, only a primitive explanation capability is provided in the synthetic reporter. Whenever a branching occurs in the synthetic reporter, the branching node is stacked in the explanation stack. If the question is given, then the stack is dumped from the top, depending on the level of the question. The level of the answer is the same as the level of the stack. The current explanation mechanism needs to be enhanced to include the reasoning base of the sleep disorder in the future system expansion.

Example of Data Processing

Some examples of the sleep data processing done by the system are given in this section. Fig. 5.8 shows the data format in each step of the frontend processor. (a) shows one of the four channel inputs (occipital channel) of 10 seconds. (b) is the same channel analog output after passing through a linear phase FIR alpha filter. Eight filters are assigned to four input channels . (c) shows an alpha wave counter output which counts the number of alpha waves using wave criteria. (d) shows an alpha wave detector output which displays a certain amount of time depending on each waveform activity. (e) shows 10 seconds of decoded sampled outputs of each waveform detector. A four hexadecimal number contains a quarter second channel information. The first 10 bits of four hexadecimal number denotes each waveform existence and the 11th bit is used for the minute indicator.

Fig. 5.9 shows the data format in each step of the main processor. (a) shows a preprocessed data format of 10 seconds of the decoder output in the channel descriptor. In the first pass analysis, the priority of the detector is resolved. The encircled number shows an artifact priority over a delta wave, which ignores the delta wave detector output if the artifact exists in the eye channel. (b) is a 20 minutes second pass analysis output of the translator. Each row denotes a different waveform activity and each column denotes one minute waveform activities. The characters "H", "M", "L", and "N" are used to denote "high", "medium", "low", and "none", respectively. (c) is the final output of the synthetic reporter. Sleep staging rules and heuristic REM detection rules are implemented in the REM module of the synthetic reporter. Using a four minute moving window REM sleep stage is detected. A minute state description shown in example is available depending on user request.

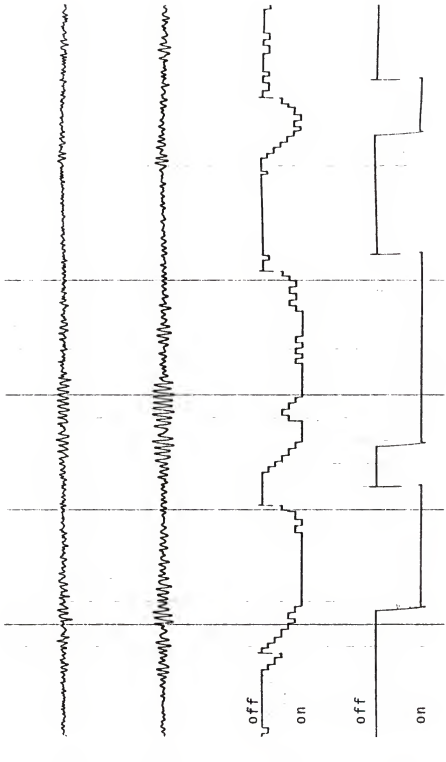


Fig. 5.8 Data Format in Token Generator

(a)
```
22A0=0C08  0C08  0E08  0E08  0E08  0E08  0E08  0A08
22B0=0A08  0A08  0808  0848  0848  0808  0A08  0A08
22C0=0A08  0A08  0A08  0A08  0A48  0A49  0A40  0A08
22D0=0848  0848  0808  0808  0C08  0C08  0A08  0A08
22E0=0A08  0A08  0A08  0A08  0A08  0E08  0E08  0E08
```

(b)
```
0C08 0C08 0600 0600 0600 0600 0600 0200
0200 0200 0200 0808 0000 0808 0208 0200
0200 0200 0200 0200 0000 0000 0000 0809
0000 0000 0808 0808 0C08 0C08 0200 0200
0200 0200 0200 0200 0200 0600 0600 0600
```

(c)
```
alpha        L H M L L N N N N N N N N N N N N N N
beta         N M L L L L N L N N N N N N N N N N N
delta        N N H H H H H H H H H H H H H H H H H
sigma        L L L L L L H L H H H H H H H H H H H
theta        N N N N N N L N N L L N N N N N N L N N
artifact     N N N N N N N N N N N N N N N N N N N
eye slow     H H N M L L L L N N L N N N N N L L N N
frontal slow N L N L L L L N L L L L L L L L M L M M
parietal "   N L N N N N N N L N N L L N L N L M M M
occipital "  N M L N N L N L L N N N N L L L L M M M
```

(d)
```
018 minute: sleep stage 4
            alpha,                      no
            beta,                       no
            delta,                      high
            sigma,                      high
            theta,                      low
            artifact,                   no
            eye slow,                   low
            frontal slow,               low
            parietal slow,              medium
            occipital slow,             medium activity

                 high delta,
                 also in previous minutes:

                      ------>   NO REM period
```

Fig. 5.9 Data Format in Token Processor

System Status

The present EEG expert system is in a prototype stage. The whole system evaluation is not possible without rule refinement. The REM period detection module is currently being tested. The intermediate results of the REM period analysis are discussed in the next chapter. It demonstrated a good result for limited data. More data have to be analyzed to refine REM period detection rules. The frontend processor, which consists of filters, detectors, and a multiplexor, is in refinement stage. The main processor, which consists of a channel descriptor and a synthetic reporter, is not complete. Although the channel descriptor is in a refinement stage, the synthetic reporter is far from complete. To complete the synthetic reporter design, the knowledge structure of the sleep disorder diagnosing rules first have to be defined and the knowledge representation techniques need to be used properly.

For the frontend processor refinement, each module, such as filter, detector, and multiplexor, needs to be tested. All the responses affect the zero crossing. In the filter design, the effects of linear phase intercept distortion, phase linearity, and skirt characteristics to zero crossing have to be studied more quantitatively and have to be considered in the filter design. Generally, the zero intercept distortion condition gives constraints to the zero positioning in the linear phase filter design. Also, the zero positioning of the linear phase filter affects the

amplitude characteristics of the filter. In the detector refinement, the waveform criteria and the amplitude criteria need to be adjusted to closely simulate visual analysis. This can only be achieved by analyzing a lot of data with different criteria. For the multiplexor, the sampling rate for token generation needs to be considered to decrease the use of on-board memory and for the compact data acquisition.

In the channel descriptor of the main processor, the signal grammar needs to be clearly defined. Although the current grammar takes any kind of waveform token, more syntactic patterns can be simply recognized by a proper grammar. This will decrease the semantic actions in the present system and can provide the capability of erroneous data handling. In the synthetic reporter, the knowledge base is currently composed of only the channel list. The connection link between the channel description list and sleep disorder is not provided yet. In the synthetic reporter design, first the task definition has to be clearly defined. Once the task is defined the synthetic reporter can be designed using a pseudo channel description list.

System expansion demands an extensive examination of the knowledge structure. As the knowledge base grows with system expansion, the knowledge representation method, inference engine, and the explanation of the reasoning base have to be restructured. For the data base management the binary data transfer and store program need to be developed in the present system. A binary file handling capability

will provide a global data pool for the expert system. For the user friendly system the controller, which provides the interactive module, control needs to be added. Currently the interactive rule management or system parameter adjustments capability is not provided. The system should be enhanced with an interactive rule management module. This module will enable the system to explain the reasoning base at any state and allow the user to easily access the knowledge base in order to add and modify the rules.

CHAPTER 6
SYSTEM EVALUATION AND CONCLUSIONS

System Evaluation

Today the system is in the prototype stage. The present system evaluation emphasis is qualitative rather than quantitative. As an initial refinement, the basic system rules were evaluated and modified corresponding to the analysis results. The rule modifications are performed on the rule files and the the system is automatically regenerated and evaluated. The simplicity of the rule modification is essential for an expert system development. The system refinements need flexible rule management and supporting tools.

Although it is difficult to check the validity and reliability of the present system as there is no standard data base with which to compare the results, the REM period detection of the present system is evaluated to show how the system performs and can be refined, by comparing the results obtained from the present system with those obtained from the visual analysis minute by minute. The results demonstrate that the analyzing method of the present system can interpret the sleep data as much in the same way as the visual analysis method and that the rule modifications are properly handled in the present system.

One subject was selected in each of the following age groups: Group1: 25 to 34 years, Group2: 43 to 53 years, Group3: 67 to 79 years. Six hours sleep data of each subject were processed off-line. A SANGAMO 3500 recoder with sixteen channels and a GRASS Model 78 polygraph system were utilized to replay and display the sleep data prerecorded on the magnetic tapes. The recoder output was connected to SAHC [Sm78]. The gain of the SAHC external output of each channel was adjusted to 1 Volt peak to peak for 50 uV EEG/EOG input. The external outputs of the SAHC were connected to EEG polygraph system and the A/D converter input of the system.

The rules used for the channel description are:

1) "N", "L", "M", and "H" denote the total corresponding activity time range (see Fig. 5.4) in an epoch.

2) The alpha activity in presence of a delta wave is not counted for the total alpha activity time calculation.

3) The slow wave on the EOG channel with a delta activity is not regarded as a REM.

4) The slow wave on the EOG channel with a slow wave on the occipital or central channel is not regarded as a REM.

5) A slow wave which has a delta wave in the previous or the following 2 seconds is not counted for the total REM activity time calculation.

6) The tokens with artifact activity are ignored.

The REM period detection rules applied to the channel description list are:

1) If alpha activity is H (larger than 30 seconds) in an epoch, it is not regarded as a REM period.

2) An epoch surrounded by H alpha minutes is not regarded as a REM period.

3) If delta activity in an epoch is equal to or greater than L (larger than 2.5 seconds), the epoch is not regarded as a REM period.

4) An epoch surrounded by M delta minutes, or H delta minutes, or M and H delta minutes is not regarded as a REM period.

5) If the slow wave activity (0.3 HZ- 6.0Hz) in an epoch is L, M or H(larger than 2.5 seconds) in occipital or parietal channel, it is not regarded as a REM period.

6) An epoch which has L, M, or H REM is regarded as a REM period with the exception of the REM period detection rules from 1 to 5.

7) A period between REM periods is also regarded as a REM period.

Fig. 6.1 shows the intermediate results of the system. The upper row and lower row denote non-REM period data and REM period data, respectively. The first element in each four tuple denotes the total non-REM or REM minutes in one hour data. The second element denotes the total correct detection minutes. The third element denotes the total missed minutes during detection. The fourth element denotes the total false detection minutes.

Table 6.1

Data Number: Hour	10071	11735	11747
1st	(60,58,2,0) (0,0,0,2)	(60,60,0,0) (0,0,0,0)	(60,59,0,0) (0,0,0,1)
2nd	(58,58,0,1) (2,1,1,0)	(60,56,4,0) (0,0,0,4)	(43,42,1,3) (17,14,3,1)
3rd	(55,55,0,3) (5,2,3,0)	(52,52,0,3) (13,10,3,0)	(45,45,0,2) (15,13,2,0)
4th	(53,53,0,3) (7,4,3,0)	(45,42,3,7) (15,8,7,3)	(55,55,0,1) (5,5,0,1)
5th	(29,18,1,4) (31,27,4,1)	(51,51,0,6) (9,3,6,0)	(34,32,2,5) (26,21,5,2)
6th	(44,44,0,3) (16,13,3,0)	(45,45,0,15) (15,0,15,0)	(53,53,0,2) (7,5,2,0)
Total for REM period	(61,47,14,3)	(52,21,31,7)	(70,58,12,5)

In the first run of the data, a 77% (10071), 40% (11735) and 83% (11747) true REM period were obtained in the 360 minutes record for each subject. In the first data analysis, the following points were observed.

1) The number of REM clusterings is mostly detected correctly in all the age groups.

2) Most REM periods not detected bu the system are due to the channel descriptor and the synthetic reporter. The rule which ignores the previous and the following 2 seconds of a delta wave and the rule which does not regard a slow wave on

EOG channel with a delta wave as a REM (channel description rule 5 and 3) incorrectly removes REM waves. The rule which does not regard a REM minute when the total delta activity time is larger than 2.5 seconds incorrectly removes REM minutes. (9 minutes, 21% in data 11735 of Group3). When a clear REM period cluster is detected, two minute REM period smoothing can decrease the number of the missed REM periods almost without increasing false alarms. One subject (11735) had more than 30 seconds alpha time with clear REM waves. In this subject, the high alpha activity (channel description rule 1) removes REM period 16 minutes (31%).

3) The optimum threshold value for each detector is not easy to determine. Once the threshold value is changed, all the tokens need to be regenerated, i.e., the data has to be run again on-line. For setting up the threshold values, four different records in different age group were utilized.

4) The results are sensitive to delta wave activities. The dropout of the tape and bad electrode contacts of (11735) generates false detections of the delta waves, which in turn removes the REM waves in the REM period detection.

The problems described previously indicate more rules need to be added and refined. For instance, the rules for the delta wave and artifact need to be changed and evaluated again utilizing the generated tokens. The channel description list can also be smoothed to get rid of false detections, i.e., a one minute false REM period detection.

The same data tokens were analyzed using different rules. All the rule changes have been made with the lex-specs or semantic action file of the corresponding channel descriptor. After changing the rules, the system is easily regenerated by executing a command file. The simplicity of the rule changes or the system regeneration gives great flexibility to the present system rule management.

Table 6.2 shows the results after removing the rules which ignore the tokens with artifact and which do not regard a slow wave on the EOG channel as a REM when there exists a slow wave on the occipital or parietal channel. Table 6.4 shows the results after changing the effective REM activity calculation (rule 5 of the channel description) range from 2 seconds to 1 second.

Table 6.2

Data Number: Hour	10071	11735	11747
1st	(60,58,2,0) (0,0,0,2)	(60,60,0,0) (0,0,0,0)	(60,59,1,0) (0,0,0,1)
2nd	(58,58,0,1) (2,1,1,0)	(60,56,4,0) (0,0,0,4)	(43,42,1,3) (17,14,3,1)
3rd	(55,55,0,2) (5,3,2,0)	(47,46,1,3) (13,10,3,1)	(45,45,0,1) (15,14,1,0)
4th	(53,53,0,3) (7,4,3,0)	(45,42,3,7) (15,8,7,3)	(55,55,0,0) (5,5,0,0)
5th	(29,28,1,4) (31,27,4,1)	(51,51,0,5) (9,4,5,0)	(34,32,2,4) (26,22,4,2)
6th	(44,44,0,2) (16,14,2,0)	(45,45,0,15) (15,0,15,0)	(53,53,0,0) (7,7,0,0)
Total for REM period	(61,49,12,3)	(52,22,30,8)	(70,62,8,4)

Table 6.3

Data Number: Hour	10071	11735	11747
1st	(60,58,2,0) (0,0,0,2)	(60,60,0,0) (0,0,0,0)	(60,59,1,0,0) (0,0,0,1)
2nd	(58,58,0,0) (2,2,0,0)	(60,56,4,0) (0,0,0,4)	(43,42,1,3) (17,14,3,1)
3rd	(55,55,0,2) (5,3,2,0)	(47,46,1,2) (13,11,2,1)	(45,45,0,1) (15,14,1,0)
4th	(53,53,0,2) (7,5,2,0)	(45,42,3,5) (15,10,5,3)	(55,54,1,0) (5,5,0,1)
5th	(29,28,1,2) (31,29,2,1)	(51,51,0,3) (9,6,3,0)	(34,32,2,4) (26,22,4,2)
6th	(44,44,0,2) (16,14,2,0)	(45,45,0,14) (15,1,14,0)	(53,53,0,0) (7,7,0,0)
Total for REM period	(61,53,8,3)	(52,28,24,8)	(70,62,8,5)

The Table 6.4 shows the results of the rule change in the system.

Table 6.4

	d10071	d11735	d11747
1st run	(61,47,14,3)	(52,21,31,7)	(70,58,12,5)
2nd run	(61,49,12,3)	(52,22,30,8)	(70,62,8,5)
3rd run	(61,53,8,3)	(52,28,24,8)	(70,62,8,5)

The results obtained from the Table 6.5 shows 87%, 54%, and 89% REM period detection. Although the results are not satisfactory, it was clearly demonstrated that the system performance level can be increased as the system rules grow and are refined.

Conclusions

The basic system frame design and REM module are completed. Although it is far from a final form, it is shown that the data driven system based on the time domain waveform detection method (Sm78) and heuristic rules for symbolic signal processing are feasible, especially for simulating the gestalt perception of low frequency biomedical signals. The experience of building an expert system has shown that software for design aids are essential for the expert system development, especially for the frequently

changing rule implementation. Further system development, particularly automatic waveform detector generation and proper knowledge base management in synthetic reporter will enhance the system to be used for biomedical signal analysis applications.

APPENDIX 1
TOKEN GENERATOR PROGRAM

The main part of the token generator program is given to show the structure of the frontend processor. The following is the TI-9900 assembly program of interrupt handler (HANDLER), alpha filter (ALPHA), alpha detector (ALPHADET), and main (FRONT). The other parts of the program (the other filters and detectors) are available in the EEG laboratory of Electrical Engineering, University of Florida.

```
PAGE: 1                                 14:26  09/14/84          PDOS ASM R2.4
                                                                  FILE: HANDLER/1,SLEEP

                                        *
                                        *   MAIN CONTROL OF DATA SAMPLING AND AQUISITION
                                        *   LOW PASS FILTERING EVERY CHANNEL WITH ZERO
                                        *   AT -1 FOR 480HZ SAMPLING RATE TO ELIMINATE
                                        *   ALIASING PROBLEM
                                        *
                                            IDT     'HANDLER'
        CFF8                    MUX     EQU     >CFF8       ;CHANNEL MULTIPLEXING
        CFFC                    EOC     EQU     >CFFC       ;END OF CONVERSION
        CFFA                    CON     EQU     >CFFA       ;A/D CONDERSION STARTER
        CFFE                    ADC     EQU     >CFFE       ;OUTPUT OF A/D CONVERTER
        F088                    AEX     EQU     >F088       ;ALPHA EXIST
        F288                    BEX     EQU     >F288       ;BETA EXIST
        F488                    DEX     EQU     >F488       ;DELTA EXIST
        F688                    SEX     EQU     >F688       ;SIGMA EXIST
        F888                    TEX     EQU     >F888       ;THETA EXIST
        FA88                    FEX     EQU     >FA88       ;ARTIFACT EXIST
        15B8                    0EX     EQU     >15B8       ;OREM EXIST
        17B8                    PEX     EQU     >17B8       ;PREM EXIST
        19B8                    QEX     EQU     >19B8       ;QREM EXIST
        1BB8                    REX     EQU     >1BB8       ;REM EXIST
        11F0                    CH0     EQU     >11F0       ;CH0 OUTPUT PORT
        11F2                    CH1     EQU     >11F2       ;CH1 OUTPUT PORT
        11F4                    CH2     EQU     >11F4       ;CH2 OUTPUT PORT
        11F6                    CH3     EQU     >11F6       ;CH3 OUTPUT PORT
        11CF                    CH00    EQU     >11CF       ;DELAY UNIT FOR CH0 LPF
        11CD                    CH11    EQU     >11CD       ;DELAY UNIT FOR CH1 LPF
        11CB                    CH22    EQU     >11CB       ;DELAY UNIT FOR CH2 LPF
        11C9                    CH33    EQU     >11C9       ;DELAY UNIT FOR CH3 LPF
        11F8                    CLOCK   EQU     >11F8       ;CLOCK
        11FA                    NDEL    EQU     >11FA       ;# OF DELTA IN A MIN.
        11FC                    NSPI    EQU     >11FC       ;# OF SPINDLE IN A MIN.
        11FE                    PSPI    EQU     >11FE       ;PREVIOUS SPINDLE CHECKER
        11E0                    AT      EQU     >11E0       ;TOTAL ALPHA TIME
        11E2                    BT      EQU     >11E2       ;TOTAL BETA TIME
        11E4                    DT      EQU     >11E4       ;TOTAL DELTA TIME
        11E6                    DN      EQU     >11E6       ;TOTAL # OF DELTA OCCURANCE
        11E8                    SN      EQU     >11E8       ;TOTAL # OF SPINDLE
        11EA                    TT      EQU     >11EA       ;TOTAL THETA TIME
        11D2                    ATM     EQU     >11D2       ;ALPHA TIMER
        11D4                    BTM     EQU     >11D4       ;BETA TIMER
        11D6                    DTM     EQU     >11D6       ;DELTA TIMER
        11D8                    DTN     EQU     >11D8       ;# OF DELTA WAVE
        11DA                    STN     EQU     >11DA       ;# OF SPINDLE
        11DC                    TTM     EQU     >11DC       ;THETA TIMER
        11DE                    FTM     EQU     >11DE       ;ARTIFACT TIMER
        DDDD                    DACD    EQU     >DDDD       ;DUMMY
                                *
        0000'FFAA                       AORG    >FFAA       ;INT LEVEL 3 VECTOR ADDRESS
FFAA: 0460 1000                         B       @>1000      ;TO HANDLER
        FFAE 1000                       AORG    >1000
                                *
                                *   CHANNEL MULTIPLEXING AND LOW PASS FILTERING
                                *     R1: CHANNEL SELECTOR
                                *
1000: 04C1                              CLR     R1          ;CLEAR CHANNEL SELECTOR
```

```
PAGE: 2                          14:26  09/14/84          PDOS ASM R2.4
                                                          FILE: HANDLER/1,SLEEP

1  1002: 04E0 CFF8                    CLR     @MUX       ;GET CH0
2  1006: C801 CFFA                    MOV     R1,@CON    ;INITIATE A/D CONVERSION
3  100A: 0581                         INC     R1         ;GET CH1
4  100C: C820 11CF 11F0               MOV     @CH00,@CH0 ;SHIFT FOR CH0 LPF
5  1012: C020 CFFC       WAIT0        MOV     @EOC,R0    ;GET EOC TEST BIT
6  1016: C000                         MOV     R0,R0      ;END OF CONVERSION?
7  1018: 15FC                         JGT     WAIT0
8  101A: C820 CFFE 11CF               MOV     @ADC,@CH00 ;PUT VALUE AT CH00
9  1020: C801 CFF8                    MOV     R1,@MUX    ;SELECT CH1
10 1024: C801 CFFA                    MOV     R1,@CON    ;INITIATE A/D CONVERSION
11 1028: 0581                         INC     R1         ;GET CH2
12 102A: A820 11CF 11F0               A       @CH00,@CH0 ;LPF OUT OF CH0
13 1030: C820 11CD 11F2               MOV     @CH11,@CH1 ;SHIFT FOR CH1 LPF
14 1036: C020 CFFC       WAIT1        MOV     @EOC,R0    ;GET EOC TEST BIT
15 103A: C000                         MOV     R0,R0      ;END OF CONVERSION?
16 103C: 15FC                         JGT     WAIT1
17 103E: C820 CFFE 11CD               MOV     @ADC,@CH11 ;PUT VALUE AT CH11
18 1044: C801 CFF8                    MOV     R1,@MUX    ;SELECT CH2
19 1048: C801 CFFA                    MOV     R1,@CON    ;INITIATE A/D CONVERSION
20 104C: 0581                         INC     R1         ;GET CH3
21 104E: A820 11CD 11F2               A       @CH11,@CH1 ;OUTPUT OF CH1
22 1054: C820 11CB 11F4               MOV     @CH22,@CH2 ;SHIFT FOR CH2 LPF
23 105A: C020 CFFC       WAIT2        MOV     @EOC,R0    ;GET EOC TEST BIT
24 105E: C000                         MOV     R0,R0      ;END OF CONVERSION?
25 1060: 15FC                         JGT     WAIT2
26 1062: C820 CFFE 11CB               MOV     @ADC,@CH22 ;PUT VALUE ATCH22
27 1068: C801 CFF8                    MOV     R1,@MUX    ;SELECT CH3
28 106C: C801 CFFA                    MOV     R1,@CON    ;INITIATE A/D CONVERSION
29 1070: A820 11CB 11F4               A       @CH22,@CH2 ;OUTPUT OF CH2
30 1076: C820 11C9 11F6               MOV     @CH33,@CH3 ;SHIFT FOR CH3
31 107C: C020 CFFC       WAIT3        MOV     @EOC,R0    ;GET EOC TEST BIT
32 1080: C000                         MOV     R0,R0      ;END OF CONVERSION
33 1082: 15FC                         JGT     WAIT3
34 1084: C820 CFFE 11C9               MOV     @ADC,@CH33 ;PUT VALUE AT CH33
35 108A: A820 11C9 11F6               A       @CH33,@CH3 ;OUTPUT OF CH3 LPF
36                              *
37                              *   MINUTE COUNT AND DATA AQUISITION
38                              *     R2: MINUTE COUNTER
39                              *     R3: STORAGE POINTER
40                              *     R4: ALPHA TIMER
41                              *     R5: BETA TIMER
42                              *     R6: DELTA TIMER
43                              *     R7: PREVIOUS DELTA CHECKER
44                              *     R8: QUARTER SEC DATA STARTING ADDR.
45                              *     R9:  DATA MULTIPLEXING REG.
46                              *     R10: QUARTET SEC. TIMER
47                              *
48 1090: 0582                         INC     R2         ;EVERY SAMPLING INTERVAL
49 1092: 058A                         INC     R10        ;QUARTER SEC TIMER
50 1094: 028A 0078                    CI      R10,120    ;QUARTER SEC ELAPSED?
51 1098: 1143                         JLT     MDATA      ;KEEP CHECKING
52 109A: 04C9                         CLR     R9         ;DATA MUX REG CLEAR
53 109C: 8282                         C       R2,R10     ;FIRST QUARTER SEC
54 109E: 1502                         JGT     NOMARK     ;IN MINUTE INTERVAL
55 10A0: 0269 1000                    ORI     R9,>1000   ;TIME MARK SET
56 10A4: 04CA            NOMARK       CLR     R10        ;RESET QUARTER SEC TIMER
```

```
                                                          PDOS ASM R2.4
PAGE: 3                          14:27  09/14/84          FILE: HANDLER/1,SLEEP

1                                *
2   10A6: C820 F088 F088                 MOV     @AEX,@AEX ;TEST ALPHA
3   10AC: 1302                           JEQ     QBX       ;IF NOT, TEST BETA
4   10AE: 0269 0800                      ORI     R9,>0800  ;SET 12TH BIT
5   10B2: C820 F288 F288 QBX             MOV     @BEX,@BEX ;TEST BETA
6   10B8: 1302                           JEQ     QDX       ;IF NOT, TEST DELTA
7   10BA: 0269 0400                      ORI     R9,>0400  ;SET 11TH BIT
8   10BE: C820 F488 F488 QDX             MOV     @DEX,@DEX ;TEST DELTA
9   10C4: 1302                           JEQ     QSX       ;IF NOT, TEST SIGMA
10  10C6: 0269 0200                      ORI     R9,>0200  ;SET 10TH BIT
11  10CA: C820 F688 F688 QSX             MOV     @SEX,@SEX ;TEST SIGMA
12  10D0: 1302                           JEQ     QTX       ;IF NOT, TEST THETA
13  10D2: 0269 0100                      ORI     R9,>0100  ;SET 9TH BIT
14  10D6: C820 F888 F888 QTX             MOV     @TEX,@TEX ;TEST THETA
15  10DC: 1302                           JEQ     QFX       ;IF NOT, TEST ARTIFACT
16  10DE: 0269 0080                      ORI     R9,>0080  ;SET 8TH BIT
17  10E2: C820 FA88 FA88 QFX             MOV     @FEX,@FEX ;TEST ARTIFACT
18  10E8: 1302                           JEQ     QOX       ;IF NOT, TEST OREM
19  10EA: 0269 0040                      ORI     R9,>0040  ;SET 6TH BIT
20  10EE: C820 15B8 15B8 QOX             MOV     @OEX,@OEX ;TEST OREM
21  10F4: 1302                           JEQ     QPX       ;IF NOT, TEST PREM
22  10F6: 0269 0008                      ORI     R9,>0008  ;SET 4TH BIT
23  10FA: C820 17B8 17B8 QPX             MOV     @PEX,@PEX ;TEST PREM
24  1100: 1302                           JEQ     QQX       ;IF NOT, TEST QREM
25  1102: 0269 0004                      ORI     R9,>0004  ;SET 3RD BIT
26  1106: C820 19B8 19B8 QQX             MOV     @QEX,@QEX ;TEST QREM
27  110C: 1302                           JEQ     QRX       ;IF NOT, TEST REM
28  110E: 0269 0002                      ORI     R9,>0002  ;SET 2ND BIT
29  1112: C820 1BB8 1BB8 QRX             MOV     @REX,@REX ;TEST REM
30  1118: 1302                           JEQ     DMUX      ;IF NOT, PROCESS MIN DATA
31  111A: 0269 0001                      ORI     R9,>0001  ;SET 1ST BIT
32  111E: CE09              DMUX         MOV     R9,*R8+   ;STORE MULTIPLEXED DATA
33  1120: 020C 0120         MDATA        LI      R12,>120  ;GET TMS 9901 PIO CRU ADDR
34  1124: 0282 03C0                      CI      R2,960    ;2 SEC INTERVAL
35  1128: 1502                           JGT     PULSE
36  112A: 1D06                           SBO     6         ;OUTPUT 1 AT PIO P6(P4-12)
37  112C: 1001                           JMP     PULSE1
38  112E: 1E06              PULSE        SBZ     6         ; OUTPUT 0 AT PIO P6(P4-12)
39  1130: 0282 7080         PULSE1       CI      R2,480*60 ;1 MINUTE ELAPSED?
40  1134: 111F                           JLT     COUNT     ;KEEP COUNTING
41  1136: 05A0 11F8                      INC     @CLOCK    ;CLOCK
42  113A: CCE0 11F8                      MOV     @CLOCK,*R3+ ;STORE CLOCK VALUE
43  113E: CCC4                           MOV     R4,*R3+   ;STORE ALPHA TIME
44  1140: CCC5                           MOV     R5,*R3+   ;STORE BETA TIME
45  1142: CCC6                           MOV     R6,*R3+   ;STORE DELTA TIME
46  1144: CCE0 11FA                      MOV     @NDEL,*R3+  ;STORE # OF DELTA OCCURANCE
47  1148: CCE0 11FC                      MOV     @NSPI,*R3+  ;STORE # OF SIGMA SPINDLE
48  114C: A804 11E0                      A       R4,@AT    ;TOTAL ALPHA TIME
49  1150: A805 11E2                      A       R5,@BT    ;TOTAL BETA TIME
50  1154: A806 11E4                      A       R6,@DT    ;TOTAL DELTA TIME
51  1158: A820 11FA 11E6                 A       @NDEL,@DN ;TOTAL # OF DELTA
52  115E: A820 11FC 11E8                 A       @NSPI,@SN ;TOTAL # OF SPINDLE
53  1164: 04C2                           CLR     R2        ;RESET MINUTE CONUTER
54  1166: 04C4                           CLR     R4        ;RESET ALPHA TIMER
55  1168: 04C5                           CLR     R5        ;RESET BETA TIMER
56  116A: 04C6                           CLR     R6        ;RESET DELTA TIMER
```

```
                                                        PDOS ASM R2.4
PAGE: 4                         14:27  09/14/84          FILE: HANDLER/1,SLEEP

1  116C: 04E0 11FA                      CLR     @NDEL       ;RESET DELTA # COUNTER
2  1170: 04E0 11FC                      CLR     @NSPI       ;RESET SIGMA # COUNTER
3  1174: C060 F088      COUNT           MOV     @AEX,R1     ;GET ALPHA EXIST
4  1178: C041                           MOV     R1,R1       ;IS THERE ALPHA?
5  117A: 1301                           JEQ     BX          ;IF NO, TEST BETA
6  117C: 0584                           INC     R4
7  117E: C060 F288      BX              MOV     @BEX,R1     ;GET BETA EXIST
8  1182: C041                           MOV     R1,R1       ;IS THERE BETA?
9  1184: 1301                           JEQ     DX          ;IF NO, TEST DELTA
10 1186: 0585                           INC     R5          ;BETA TIME
11 1188: C060 F488      DX              MOV     @DEX,R1     ;GET DELTA EXIST
12 118C: C041                           MOV     R1,R1       ;IS THERE DELTA?
13 118E: 1305                           JEQ     NODEX       ;IF NOT
14 1190: 0586                           INC     R6          ;INCREASE DELTA TIME
15 1192: C1C7                           MOV     R7,R7       ;DELTA EXISTED PREVIOUSLY?
16 1194: 1603                           JNE     SX          ;IF SO, TEST SIGMA
17 1196: 05A0 11FA                      INC     @NDEL       ;OR INCREASE DELTA COUNTER
18 119A: C1C1        NODEX              MOV     R1,R7       ;RECORD EXISTENCE OF DELTA
19 119C: C060 F688   SX                 MOV     @SEX,R1     ;GET SIGMA EXIST
20 11A0: C041                           MOV     R1,R1       ;IS THERE A SIGMA?
21 11A2: 1306                           JEQ     NOSEX       ;IF NOT
22 11A4: C820 11FE 11FE                 MOV     @PSPI,@PSPI ;SIGMA EXISTED PREVIOUSLY?
23 11AA: 1604                           JNE     GO          ;IF PREVIOUSLY EXISTED
24 11AC: 05A0 11FC                      INC     @NSPI       ;RECORD SIGMA OCCURANCE
25 11B0: C801 11FE   NOSEX              MOV     R1,@PSPI    ;RECORD SIGMA EXISTENCE
26                                   *  RESET INTERRUPT MECHANISM
27 11B4: 020C 0100   GO                 LI      R12,>100    ;PSI CRU BASE ADDR. SET
28 11B8: 1E00                           SBZ     0           ;SET 9901 IN INTERRUPT MODE
29 11BA: 1D03                           SBO     3           ;ENABLE INT3 IN 9901
30 11BC: 0300 0003                      LIMI    3           ;ENABLE INT3 IN 9900
31 11C0: 05CE                           INCT    R14         ;SUPPORT 'JMP $' INSTRUCTION
32 11C2: 0380                           RTWP
33                                      LINK    ALPHA/1
34                                      IDT     'ALPHA'
35                                   *  ALL THE FILTERS HAVE DUMMY OUTPUT PORT @DDDD
36                                   *  @CFF0, @CFF2 ARE USED AS OUTPUT PORTS
37                                   *ADC    EQU     >CFFE       ;SIGNAL INPUT PORT ADDR.
38         F082                      ALPHA1  EQU     >F082       ;ALPHADET R1 PRESENT INPUT
39         F084                      ALPHA2  EQU     >F084       ;ALPHADET R2 PREVIOUS INPUT
40         F1B4                      ALPOUT  EQU     >F1B4       ;ALPD OUTPUT, R10 OF ALPD
41                                   *
42                 11C4 F020                AORG    >F020
43 F020: C060 F1B4                       MOV     @ALPOUT,R1
44 F024: 0811                            SRA     R1,1        ;MAX GAIN 5, ADJUST BY 1/4
45 F026: 6181                            S       R1,R6
46 F028: A206                            A       R6,R8
47 F02A: C287                            MOV     R7,R10
48 F02C: 081A                            SRA     R10,1
49 F02E: A20A                            A       R10,R8
50 F030: A248                            A       R8,R9
51 F032: 0829                            SRA     R9,2
52 F034: C809 DDDD                       MOV     R9,@DACD
53 F038: C820 F082 F084                  MOV     @ALPHA1,@ALPHA2  ;STORE PREVIOUS VALUE
54 F03E: C809 F082                       MOV     R9,@ALPHA1  ;STORE PRESENT VALUE
55                                   *
56 F042: C248                            MOV     R8,R9
```

```
PAGE: 5                                 14:27  09/14/84                 PDOS ASM R2.4
                                                                        FILE: ALPHA/1,SLEEP

1  F044: C207                                   MOV     R7,R8
2  F046: C1C6                                   MOV     R6,R7
3  F048: C185                                   MOV     R5,R6
4  F04A: C144                                   MOV     R4,R5
5  F04C: C103                                   MOV     R3,R4
6  F04E: C0C2                                   MOV     R2,R3
7  F050: C081                                   MOV     R1,R2
8  F052: 0380                                   RTWP
9                                               LINK    ALPHADET/1
10                                              IDT     'ALPHADET'
11       F054 F0A0                              AORG    >F0A0
12                                      *
13                                      *EACH REGISTERS DENOTE FOR
14                                      *       PRESENT INPUT: R1
15                                      *       PREVIOUS INPUT: R2
16                                      *       EXIST: R4
17                                      *       HISTO: R5
18                                      *       MESH : R6
19                                      *       TIMER: R7
20                                      *       PEAK : R8
21                                      *       DTIMER: R9      :DELAY TIMER FOR DISPLAY
22                                      *
23                                      *PARALLEL I/O PORT P0(P4-20) FOR DETECTOR OUTPUT
24                                      *AVERAGE WAVE TIME 6 CONSECUTIVE WAVES IS USED
25                                      *FOR DETECTOR CRITERION
26       F070                           A1      EQU     >F070
27       F072                           A2      EQU     >F072
28       F074                           A3      EQU     >F074
29       F076                           A4      EQU     >F076
30       F078                           A5      EQU     >F078
31       F07A                           A6      EQU     >F07A
32 F0A0: 00FF                           ONE     DATA    >FF             ;MAGNIFIED FOR DAC OUT
33 F0A2: 020C 0120                              LI      R12,>120        ;GET TMS 9901 PIO CRU ADDR.
34 F0A6: 0609                                   DEC     R9              ;DTIMER ADJUSTMENT
35                                      *
36                                      *       ZERO CROSSING CHECK
37 F0A8: C041                                   MOV     R1,R1           ;WAVE(NT)>0?
38 F0AA: 135D                                   JEQ     ANOCRO          ;NO CROSSING
39 F0AC: 115C                                   JLT     ANOCRO          ;NO CROSSING, NEGATIVE GOING
40 F0AE: C082                                   MOV     R2,R2           ;WAVE((N-1)T)<0?
41 F0B0: 155A                                   JGT     ANOCRO          ;IF NOT, NO CROSSING
42                                      *
43                                      *       AVERAGE WAVE DURAION CHECK
44 F0B2: C820 F078 F07A                         MOV     @A5,@A6
45 F0B8: C820 F076 F078                         MOV     @A4,@A5
46 F0BE: C820 F074 F076                         MOV     @A3,@A4
47 F0C4: C820 F072 F074                         MOV     @A2,@A3
48 F0CA: C820 F070 F072                         MOV     @A1,@A2
49 F0D0: C807 F070                              MOV     R7,@A1
50                                      *
51                                      *       MINIMUM AMPLITUDE CHECK
52 F0D4: 0288 0030                              CI      R8,48           ;LARGER THEN MIN INPUT?
53 F0D8: 110A                                   JLT     ASENDO          ;IF NOT, NO WAVE
54                                      *
55                                      *       HISTORY CHECK: ZERO CROSSING AFTER OVER DURATION
56 F0DA: C145                                   MOV     R5,R5           ;1 MEANS HAVING HISTORY
```

```
PAGE: 6                                 14:27  09/14/84                 PDOS ASM R2.4
                                                                        FILE: ALPHADET/1,SLEEP

1  F0DC: 1607                                   JNE     ASENDH     ;IF IT HAS HISTORY, SEND 0 TO
2                                          *                      ;MESH AND SET HISTORY 0
3                                          *
4                                          *   WAVE LENGTH(DURATION) CHECK
5  F0DE: 0287 0009                              CI      R7,9       ;LARGER THEN MIN INTERVAL?
6  F0E2: 1A05                                   JL      ASEND0     ;IF NO WAVE, SEND 0 TO MESH
7  F0E4: 0287 000F                              CI      R7,15      ;SMALLER THEN MAX INTERVAL?
8  F0E8: 1B02                                   JH      ASEND0     ;IF NOT, SEND 0 TO MESH
9  F0EA: 1008                                   JMP     ASEND1     ;IF SO, SEND 1 TO MESH
10                                         *
11                                         *   HISTORY VALUE RESET
12 F0EC: 04C5                              ASENDH  CLR     R5         ;SET HISTO TO 0
13                                         *
14                                         *   SEND 0 TO MESH THERE IS NO WAVE
15 F0EE: 0286 0003                         ASEND0  CI      R6,3       ;MESH > 3?
16 F0F2: 1202                                      JLE     AMESH0     ;MESH<4?, THEN SET MESH TO 0
17 F0F4: 0606                                      DEC     R6         ;MESH>3?, THEN MESH:=MESH-1
18 F0F6: 1006                                      JMP     AWAVE
19 F0F8: 04C6                              AMESH0  CLR     R6         ;SET MESH TO 0
20 F0FA: 100E                                      JMP     AOUT0
21                                         *
22                                         *   SEND 1 TO MESH IF THERE IS A SATISFIED WAVE
23 F0FC: 0286 0006                         ASEND1  CI      R6,6       ;REACHES MAXIMUM # OF WAVE?
24 F100: 1401                                      JHE     AWAVE      ;IF SO, KEEP MAX VALUE
25 F102: 0586                                      INC     R6         ;IF NOT, MESH:=MESH+1
26                                         *
27                                         *   IS THERE A WAVE WITH PREVIOUS OUTPUT?
28 F104: 0284 00FF                         AWAVE   CI      R4,>FF     ;WAS THERE A WAVE:EXIST=FF?
29 F108: 1A04                                      JL      ANONEX     ;IF NOT
30                                         *
31                                         *   WHEN THERE WAS A WAVE
32 F10A: 0286 0004                                 CI      R6,4       ;CHECK HYSTERESIS CONDITION
33 F10E: 1409                                      JHE     AOUT1      ;IF SATTISFIED,SET EXIST TO 1
34 F110: 1003                                      JMP     AOUT0      ;IF NOT SET EXIST TO 0
35                                         *
36                                         *   WHEN THERE WAS NO WAVE PREVIOUSLEY
37 F112: 0286 0006                         ANONEX  CI      R6,6       ;MESH>=WAVEIN?
38 F116: 1405                                      JHE     AOUT1      ;IF SATISFIED, SET EXIST TO 1
39 F118: C249                              AOUT0   MOV     R9,R9      ;DTIMER ON?
40 F11A: 151B                                      JGT     ASETC0     ;THEN CONTINUE TO DISPLAY
41 F11C: 04C4                                      CLR     R4         ;SET EXIST TO 0
42 F11E: 1E00                                      SBZ     0          ;OUTPUT 0 TO P0(P4-20)
43 F120: 1018                                      JMP     ASETC0     ;
44 F122: 04C3                              AOUT1   CLR     R3
45 F124: A0E0 F070                                 A       @A1,R3
46 F128: A0E0 F072                                 A       @A2,R3
47 F12C: A0E0 F074                                 A       @A3,R3
48 F130: A0E0 F076                                 A       @A4,R3
49 F134: A0E0 F078                                 A       @A5,R3
50 F138: A0E0 F07A                                 A       @A6,R3
51 F13C: 0283 0038                                 CI      R3,56
52 F140: 1AEB                                      JL      AOUT0
53 F142: 0283 0054                                 CI      R3,84
54 F146: 1BE8                                      JH      AOUT0
55 F148: C120 F0A0                                 MOV     @ONE,R4    ;SET EXIST TO 1
56 F14C: 0209 0038                                 LI      R9,56      ;SET MIN ALPFA TIME
```

```
                                                        PDOS ASM R2.4
PAGE: 7                         14:27  09/14/84          FILE: ALPHADET/1,SLEEP

1  F150: 1D00                           SBO     0          ;OUTPUT 1 TO P0(P4-20)
2  F152: C804 DDDD              ASETC0  MOV     R4,@DACD   ;OUTPUT EXIST TO DAC2
3  F156: C0C6                           MOV     R6,R3      ;GET MESH VALUE AT R3
4  F158: 0A73                           SLA     R3,7       ;MESH*128
5  F15A: C803 DDDD                      MOV     R3,@DACD   ;OUTPUT OF MESH*128
6  F15E: 04C7                           CLR     R7         ;INITIALIZE TIMER
7  F160: 04C8                           CLR     R8         ;SET PEAK 0
8  F162: 0460 F184                      B       @AFINE
9                               *
10                              *   WHEN THERE IS NO ZERO CROSSING
11 F166: 0287 000F              ANOCRO  CI      R7,15      ;CHECK WAVE LENGTH (DURATION)
12 F16A: 1B08                           JH      AOVERT     ;IF SO
13 F16C: 0587                           INC     R7         ;TIMER:=TIMER+1
14                              *
15                              *   PEAK VALUE CHANGE WHEN NOT IN OVER TIME
16 F16E: C281                           MOV     R1,R10     ;GET INPUT AT R10
17 F170: 074A                           ABS     R10        ;GET ABSOLUTE VALUE
18 F172: 8288                           C       R8,R10     ;PEAK VALUE > ABSOLUTE INPUT
19 F174: 1507                           JGT     AFINE      ;THEN NO CHANGE
20 F176: C20A                           MOV     R10,R8     ;GET NEW PEAK VALUE
21 F178: 0460 F184                      B       @AFINE
22                              *
23                              *   WHEN WAVE LENGTH EXCEEDS MAX TIME
24 F17C: C160 F0A0              AOVERT  MOV     @ONE,R5    ;SET HISTO TO 1
25 F180: 0460 F0EE                      B       @ASEND0
26 F184: 0380                   AFINE   RTWP
27                                      LINK    ALPD/1
28                                      IDT     'ALPD'
29       F1B4                   ALPOUT  EQU     >F1B4      ;R10 OF ALPFW
30       F186 F1C0                      AORG    >F1C0
31 F1C0: F1A0 F1C4              ALPD    DATA    >F1A0,ALPD1
32 F1C4: C060 11F4              ALPD1   MOV     @CH2,R1
33 F1C8: 0821                           SRA     R1,2       ;ADJUST INPUT
34 F1CA: A0C1                           A       R1,R3
35 F1CC: A143                           A       R3,R5
36 F1CE: A1C5                           A       R5,R7
37 F1D0: A1C6                           A       R6,R7
38 F1D2: A207                           A       R7,R8
39 F1D4: 0828                           SRA     R8,2       ;ADJUST FILTER GAIN
40 F1D6: C808 DDDD                      MOV     R8,@DACD   ;DUMMY OUTPUT
41 F1DA: C808 F1B4                      MOV     R8,@ALPOUT
42 F1DE: C207                           MOV     R7,R8
43 F1E0: C1C6                           MOV     R6,R7
44 F1E2: C185                           MOV     R5,R6
45 F1E4: C144                           MOV     R4,R5
46 F1E6: C103                           MOV     R3,R4
47 F1E8: C0C2                           MOV     R2,R3
48 F1EA: C081                           MOV     R1,R2
49 F1EC: 0380                           RTWP
```

PAGE: 30 14:29 09/14/84 PDOS ASM R2.4
FILE: FRONT/1,SLEEP

```
36                          LINK    FRONT/1
37                          IDT     'FRONT'
38      F000        ALPHAW  EQU     >F000     ;ALPHA FILTER WORK SPACE
39      F020        ALPHA   EQU     >F020
40      F080        ADETW   EQU     >F080
41      F0A0        ADET    EQU     >F0A0
42      F200        BETAW   EQU     >F200
43      F220        BETA    EQU     >F220
44      F280        BDETW   EQU     >F280
45      F2A0        BDET    EQU     >F2A0
46      FF98        HQTIME  EQU     >FF98     ;QUARTER SEC TIMER
47      FF9E        HMTIME  EQU     >FF9E     ;MINUTE TIMER IN HANDLER
48      F08C        AMESH   EQU     >F08C     ;ALFA DETECTOR MESH R6
49      F28C        BMESH   EQU     >F28C     ;BETA DETECTOR MESH
50      F48C        DMESH   EQU     >F48C     ;DELTA MESH
51      F68C        SMESH   EQU     >F68C     ;SIGMA MESH
52      F88C        TMESH   EQU     >F88C     ;THETA MESH
53      FA8C        FMESH   EQU     >FA8C     ;ARTIFACT MESH
54      15BC        OMESH   EQU     >15BC     ;OREM MESH
55      17BC        PMESH   EQU     >17BC     ;PREM MESH
56      19BC        QMESH   EQU     >19BC     ;QREM MESH
```

```
                                                        PDOS ASM R2.4
PAGE: 31                        14:29  09/14/84          FILE: FRONT/1,SLEEP

1         1BBC          RMESH   EQU     >1BBC     ;REM MESH
2         F492          DISPL   EQU     >F492     ;DELTA DISPLAYER,S TIMER
3         11F8          CLOCK   EQU     >11F8     ;MINUTE CLOCK
4         118E          MCLOCK  EQU     >118E     ;MINUTE COUNTER
5                       *
6         1A72 1220             AORG    >1220
7  1220: 02E0 1200              LWPI    >1200
8  1224: 04E0 FF9B              CLR     @HQTIME   ;1/4 SEC HANDLER TIMER
9  1228: 04E0 FF9E              CLR     @HMTIME   ;HANDLER MINUTE TIMER
10 122C: 04E0 F08C              CLR     @AMESH    ;INITIALIZE ALFA MESH
11 1230: 04E0 F28C              CLR     @BMESH    ;INITIALIZE BETA MESH
12 1234: 04E0 F48C              CLR     @DMESH    ;INITIALIZE DELTA MESH
13 1238: 04E0 F68C              CLR     @SMESH    ;INITIALIZE SIGMA MESH
14 123C: 04E0 F88C              CLR     @TMESH    ;INITIALIZE THETA MESH
15 1240: 04E0 FA8C              CLR     @FMESH    ;ARTIFACT MESH
16 1244: 04E0 15BC              CLR     @OMESH    ;
17 1248: 04E0 17BC              CLR     @PMESH    ;
18 124C: 04E0 19BC              CLR     @QMESH    ;
19 1250: 04E0 1BBC              CLR     @RMESH    ;
20 1254: 04E0 F492              CLR     @DISPL    ;INITIALIZE DELTA TIMER
21 1258: 04E0 11D2              CLR     @ATM      ;ALPHA TIMER
22 125C: 04E0 11D4              CLR     @BTM      ;BETA TIMER
23 1260: 04E0 11D6              CLR     @DTM      ;DELTA TIMER
24 1264: 04E0 11D8              CLR     @DTN      ;# OF DELTA
25 1268: 04E0 11DA              CLR     @STN      ;# OF SPINDLE
26 126C: 04E0 11E0              CLR     @AT       ;TOTAL ALPHA TIME
27 1270: 04E0 11E2              CLR     @BT       ;TOTAL BETA TIME
28 1274: 04E0 11E4              CLR     @DT       ;TOTAL DELTA TIME
29 1278: 04E0 11E6              CLR     @DN       ;TOTAL #OF DELTA
30 127C: 04E0 11E8              CLR     @SN       ;TOTAL # OF SPINDLE
31 1280: 04E0 11F8              CLR     @CLOCK    ;INITIALIZE MINUTE CLOCK
32 1284: 04E0 118E              CLR     @MCLOCK   ;INITIALIZE MINUTE COUNTER
33 1288: 0201 CFF0              LI      R1,>CFF0  ;GET OUTPUT PORT ADDRESS
34 128C: 04E1 0006              CLR     @6(1)     ;SET GAIN 1
35 1290: 04E1 0008              CLR     @8(1)     ;DISABLE MUX AUTOINC MODE
36 1294: 0201 0100              LI      R1,>100
37 1298: C801 FFA2              MOV     R1,@>FFA2 ;HANDLER CRU ADDRESS SET
38 129C: 0201 2000              LI      R1,>2000  ;MULTIPLEXED DATA
39 12A0: C801 FF9A              MOV     R1,@>FF9A ;R8 IN HANDLER, STARTING ADDR.
40 12A4: 0201 8300              LI      R1,>8300  ;STORAGE ADDRESS
41 12A8: C801 FF90              MOV     R1,@>FF90 ;R3 IN HANDLER, STARTING ADDR.
42 12AC: 0201 F000              LI      R1,ALPHAW ;GET SUBROUTINE ALPHA WP
43 12B0: 0202 F020              LI      R2,ALPHA  ;GET SUBROUTINE ALPHA PC
44 12B4: 0203 F080              LI      R3,ADETW  ;GET ADET WORK SPACE ADDRESS
45 12B8: 0204 F0A0              LI      R4,ADET   ;GET DETECTOR ADDRESS
46 12BC: 0207 F200              LI      R7,BETAW  ;
47 12C0: 0208 F220              LI      R8,BETA   ;
48 12C4: 0209 F280              LI      R9,BDETW  ;
49 12C8: 020A F2A0              LI      R10,BDET  ;
50 12CC: 020C 0100              LI      R12,>100  ;CRU BASE ADDRESS TO PSI
51 12D0: 1E00                   SBZ     0         ;INABLE INTERRUPT
52 12D2: 1D03                   SBO     3         ;SET INT3
53 12D4: 0300 0003              LIMI    3         ;SET INT3 ON CPU
54 12D8: 0200 00C3              LI      R0,>C3    ;INITIALIZE COUNTER
55 12DC: 33C0                   LDCR    R0,15     ;START COUNTER
56 12DE: 10FF           BACK    JMP     $         ;WAIT FOR INTERRUPT 1ST
```

```
PAGE: 32                        14:29  09/14/84          PDOS ASM R2.4
                                                         FILE: FRONT/1,SLEEP

 1  12E0: 0407                         BLWP   7
 2  12E2: 0409                         BLWP   9
 3  12E4: 0420 1770                    BLWP   @PLPFF
 4  12E8: 0420 17D0                    BLWP   @PREMD
 5  12EC: 0420 1970                    BLWP   @QLPFF
 6  12F0: 0420 19D0                    BLWP   @QREMD
 7  12F4: 0420 1B70                    BLWP   @RLPFF
 8  12F8: 0420 1BD0                    BLWP   @REMD
 9  12FC: 10FF                         JMP    $          ;2ND
10  12FE: 0420 F1C0                    BLWP   @ALPD
11  1302: 0420 F3C0                    BLWP   @LPD
12  1306: 0420 F5A0                    BLWP   @LOW
13  130A: 0420 F4A0                    BLWP   @DELD
14  130E: 0420 F620                    BLWP   @SIG
15  1312: 0420 F6A0                    BLWP   @SIGD
16  1316: 0420 FA20                    BLWP   @ARTIF
17  131A: 0420 FAA0                    BLWP   @ARTD
18  131E: 10FF                         JMP    $          ;3RD
19  1320: 0401                         BLWP   1
20  1322: 0403                         BLWP   3
21  1324: 0407                         BLWP   7
22  1326: 0409                         BLWP   9
23  1328: 0420 F820                    BLWP   @THETA
24  132C: 0420 F8A0                    BLWP   @THED
25  1330: 0420 1570                    BLWP   @OLPFF
26  1334: 0420 15D0                    BLWP   @OREMD
27  1338: 10FF                         JMP    $          ;4TH
28  133A: 0420 F1C0                    BLWP   @ALPD
29  133E: 0420 F3C0                    BLWP   @LPD
30  1342: 0420 F620                    BLWP   @SIG
31  1346: 0420 F6A0                    BLWP   @SIGD
32  134A: 0420 FA20                    BLWP   @ARTIF
33  134E: 0420 FAA0                    BLWP   @ARTD
34  1352: 10FF                         JMP    $          ;5TH
35  1354: 0407                         BLWP   7
36  1356: 0409                         BLWP   9
37  1358: 0420 1770                    BLWP   @PLPFF
38  135C: 0420 17D0                    BLWP   @PREMD
39  1360: 0420 1970                    BLWP   @QLPFF
40  1364: 0420 19D0                    BLWP   @QREMD
41  1368: 0420 1B70                    BLWP   @RLPFF
42  136C: 0420 1BD0                    BLWP   @REMD
43  1370: 10FF                         JMP    $          ;6TH
44  1372: 0420 F1C0                    BLWP   @ALPD
45  1376: 0420 F3C0                    BLWP   @LPD
46  137A: 0420 F620                    BLWP   @SIG
47  137E: 0420 F6A0                    BLWP   @SIGD
48  1382: 0420 FA20                    BLWP   @ARTIF
49  1386: 0420 FAA0                    BLWP   @ARTD
50  138A: 1001                         JMP    BRIDGE
51  138C: 10A8             POST        JMP    BACK
52  138E: 10FF             BRIDGE      JMP    $          ;7TH
53  1390: 0401                         BLWP   1
54  1392: 0403                         BLWP   3
55  1394: 0407                         BLWP   7
56  1396: 0409                         BLWP   9
```

```
PAGE: 33                        14:29  09/14/84          PDOS ASM R2.4
                                                         FILE: FRONT/1,SLEEP

1   1398: 0420 F820             BLWP    @THETA
2   139C: 0420 F8A0             BLWP    @THED
3   13A0: 0420 1570             BLWP    @OLPFF
4   13A4: 0420 15D0             BLWP    @OREMD
5   13A8: 10FF                  JMP     $           ;8TH
6   13AA: 0420 F1C0             BLWP    @ALPD
7   13AE: 0420 F3C0             BLWP    @LPD
8   13B2: 0420 F5A0             BLWP    @LOW
9   13B6: 0420 F4A0             BLWP    @DELD
10  13BA: 0420 F620             BLWP    @SIG
11  13BE: 0420 F6A0             BLWP    @SIGD
12  13C2: 0420 FA20             BLWP    @ARTIF
13  13C6: 0420 FAA0             BLWP    @ARTD
14  13CA: 10FF                  JMP     $           ;9TH
15  13CC: 0407                  BLWP    7
16  13CE: 0409                  BLWP    9
17  13D0: 0420 1770             BLWP    @PLPFF
18  13D4: 0420 17D0             BLWP    @PREMD
19  13D8: 0420 1970             BLWP    @QLPFF
20  13DC: 0420 19D0             BLWP    @QREMD
21  13E0: 0420 1B70             BLWP    @RLPFF
22  13E4: 0420 1BD0             BLWP    @REMD
23  13E8: 10FF                  JMP     $           ;10TH
24  13EA: 0420 F1C0             BLWP    @ALPD
25  13EE: 0420 F3C0             BLWP    @LPD
26  13F2: 0420 F620             BLWP    @SIG
27  13F6: 0420 F6A0             BLWP    @SIGD
28  13FA: 0420 FA20             BLWP    @ARTIF
29  13FE: 0420 FAA0             BLWP    @ARTD
30  1402: 10FF                  JMP     $           ;11TH
31  1404: 0401                  BLWP    1
32  1406: 0403                  BLWP    3
33  1408: 0407                  BLWP    7
34  140A: 0409                  BLWP    9
35  140C: 0420 F820             BLWP    @THETA
36  1410: 0420 F8A0             BLWP    @THED
37  1414: 0420 1570             BLWP    @OLPFF
38  1418: 0420 15D0             BLWP    @OREMD
39  141C: 10FF                  JMP     $           ;12TH
40  141E: 0420 F1C0             BLWP    @ALPD
41  1422: 0420 F3C0             BLWP    @LPD
42  1426: 0420 F620             BLWP    @SIG
43  142A: 0420 F6A0             BLWP    @SIGD
44  142E: 0420 FA20             BLWP    @ARTIF
45  1432: 0420 FAA0             BLWP    @ARTD
46  1436: 10AA                  JMP     POST        ;JUMP LIMIT
47  1438:                       END
```

APPENDIX 2
LINEAR FIR FILTER DESIGN AND IMPLEMENTATION

An alpha filter illustrates the filter design algorithm used in the present system. Other filter functions are also given with the sampling rates.

Linear FIR Filter Design

1) Calculate the number of zeros using the following equation.

fs = N * B

where fs : sampling frequency,

N : required number of zeros of the filter,

B : required passband.

2) Place the zeros on the unit circle of the Z-plane.

3) Reduce the stopband gain by placing zeros in the stopband.

4) Analyze the filter response using DINAP [Ba78].

5) If necessary, add more zeros or adjust the zero positions to make the filter transfer function simple and then go to step 4.

Linear Filter Implementation

Ex) Alpha Filter

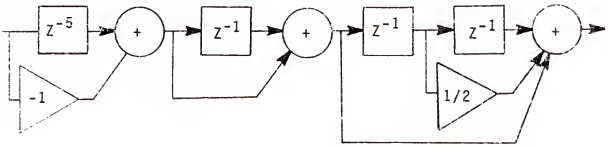
Passband : 5.0 - 24.0 Hz

Zeros : 0, 2/5, 4/5, 6/5, 8/5 , 2/3, 4/3, 1 pi

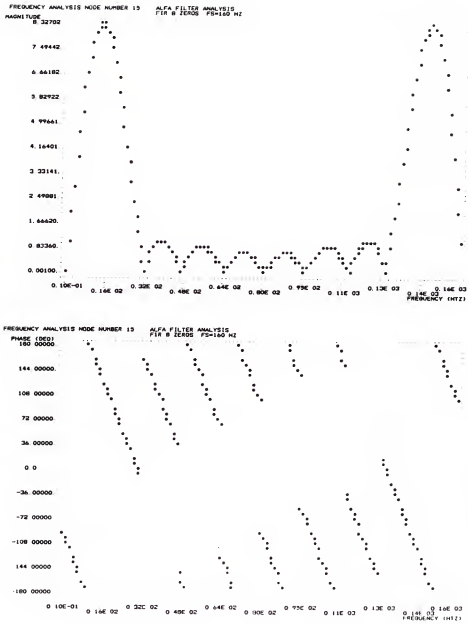
Transfer Function :

$$(1 + Z^{-5})(-1 + Z^{-1})(1 + Z/2 + Z^{-2})$$

Sampling Rate : 160 Hz



Working registers are used as shift registers.



Alpha filter frequency characteristic.

APPENDIX 3
TOKEN PROCESSOR FILES

The following are the parts of the main processor program. Each name of the file represents the functional name of the token processor. The 'pdpro.*' is the program for the translator of the main processor and the others are for the channel descriptor. The token translator, alpha (alpha.*) and REM (rslow.*) descriptor are given in order. 'Makefile' is a kind of command file which executes a series of shell commands in the UNIX operarting system. 'lexspecs', 'gram.g', 'semaction.c' are the input files for the lexical analyzer and parser generator. These files are used for the signal language compiler specification. Most of the signal understanding rules (channel descriptor) are in these files.

*** pdpro.Makefile ***

```
FILES=main.o lex.yy.o parser.o

firstpass: $(FILES)
        cc -o firstpass $(FILES) -ll
main.o: main.c
        cc -c main.c
lex.yy.o: lex.yy.c
        cc -c lex.yy.c
lex.yy.c: lexspecs
        lex lexspecs
parser.o: parser.c
        cc -c parser.c
```

*** pdpro.lexspecs ***

```
%{
#include <stdio.h>
int linenum=0;
int datanum=0;
extern int listing;
%}

HEXNUM    [A-Fa-f0-9]     /* hex notation */
ALPHA     [8-9A-Fa-f]     /* 1st bit "8" of 2nd nibble */
BETA      [4-7C-Fc-f]     /* 2nd bit "4" of 2nd nibble */
DELTA     [2367ABabEFeF]  /* 3rd bit "2" of 2nd nibble */
SIGMA     [13579BbDdFf]   /* 4th bit "1" of 2nd nibble */
THETA     [8-9A-Fa-f]     /* 1st bit "8" of 3rd nibble */
ARTIFACT  [4-7C-Fc-f]     /* 2nd bit "4" of 3rd nibble */
REM       [8-9A-Fa-f]     /* 1st bit "8" of 4th nibble */
OREM      [4-7C-Fc-f]     /* 2nd bit "4" of 4th nibble */
PREM      [2367AaBbEeFf]  /* 3rd bit "2" of 4th nibble */
QREM      [13579BbDdFf]   /* 4th bit "1" of 4th nibble */

%%
^.*\n     {linenum += 1;
        if(listing) printf("[%05d]%s",linenum,yytext);
        REJECT;
        }
[A-Fa-f0-9]{4,4}(=) ; /* address, ignore      */
[01][A-Fa-f0-9][4-7C-Fc-f][A-Fa-f0-9] {
        /* remove artifact */
        yytext[1] = '0';
        yytext[2] = '4';
        yytext[3] = '0';
        REJECT;
        }
```

```
[01][2367ABabEFef][A-Fa-f0-9][A-Fa-f0-9] {
        /* alpha with delta */
        switch(yytext[1])       {
        case 'A':
        case 'a': yytext[1] = '2'; break;
        case 'B':
        case 'b': yytext[1] = '3'; break;
        case 'E':
        case 'e': yytext[1] = '6'; break;
        case 'F':
        case 'f': yytext[1] = '7'; break;
        default : break;
        }
        REJECT;
        }
[01][2367ABEFabef][A-Fa-f0-9][89ABCDEFabcdef] {
        /* rem with delta */
        switch(yytext[3])       {
        case '8': yytext[3] = '0'; break;
        case '9': yytext[3] = '1'; break;
        case 'A':
        case 'a': yytext[3] = '2'; break;
        case 'B':
        case 'b': yytext[3] = '3'; break;
        case 'E':
        case 'e': yytext[3] = '6'; break;
        case 'F':
        case 'f': yytext[3] = '7'; break;
        default : break;
        }
        REJECT;
        }
[01][A-Fa-f0-9][A-Fa-f0-9][89ABCDEFabcdef] {
        /* rem with slow wave */
        switch(yytext[3])       {
        case 'A':
        case 'a': yytext[3] = '2'; break;
        case 'B':
        case 'b': yytext[3] = '3'; break;
        case 'E':
        case 'e': yytext[3] = '6'; break;
        case 'F':
        case 'f': yytext[3] = '7'; break;
        default : break; }
        REJECT;
        }
[A-Fa-f0-9]{4,4} {
        datanum++;
        return(1);      /* octal number        */
        }
" "      ; /* white space, ignore */
\t       ; /* white space, ignore */
\n       ; /* and count the number of columns */
.       {
```

```
        printf("Scanner error in line %d:");
        printf(" Illegal Character:",linenum);
        if(yytext[0]>=' ')
                printf("%c\n",yytext[0]);
        else
                printf("ascii %03o0,yytext[0]);
        }
```

*** pdpro.parser.c ***

```
extern char yytext[];
extern int linenum;
extern int datanum;

parser()
{
        int tok;

        while((tok=yylex())!=0) {
                printf("%s ",yytext);
                if (datanum%8 == 0)
                        printf("0);
        }
}
```

*** pdpro.main.c ***

```
#include <stdio.h>

int listing=0, trace=0;

FILE *objf;
static char *objname="asm.inp";

main(argc,argv)
int argc;
char *argv[];
{
        int i;

        for(i=1;i<argc&&argv[i][0]=='-';i++) {
                switch(argv[i][1]) {
                case 'l':
                        listing=1;
                        break;
                case 't':
                        trace=1;
                        break;
```

```
                case 'o':
                        objname=argv[++i];
                        break;
                case 'd':
                        objname=0;
                        break;
                }
        }

        if(i<argc) {
                fclose(stdin);
                if(fopen(argv[i],"r")==NULL) {
                printf("%s error: Cannot open %s0
                        ,argv[0],argv[i]);
                        exit(1);
                }
        }

        parser();
        exit(0);
}
```

*** alpha.Makefile ***

```
files=main.o lex.yy.o parser.o ptables.o semaction.o

alpha: $(files)
        cc -o alpha $(files) -ll
lex.yy.o: lex.yy.c
        cc -c lex.yy.c
lex.yy.c: lexspecs
        lex lexspecs
main.o: main.c
        cc -c main.c
parser.o: parser.c ptables.h
        cc -c parser.c
ptables.o: ptables.c ptables.h
        cc -c ptables.c
semaction.o: semaction.c
        cc -c semaction.c
ptables.c: gram.g
        slrgen -u gram.g ptables.c
```

*** alpha.lexspecs ***

```
%{
#include <stdio.h>
int linenum=0;
int datanum=0;
int alphatime=0;
```

```
int minute=0;
extern int listing;
%}

DIGIT           [01]
HEXNUM          [0-9A-Fa-f]
ALPHA           [89A-Fa-f]
NOALPHA         [0-7]
BETA            [4-7C-Fc-f]
DELTA           [2367ABEFabef]
SIGMA           [13579BDFbdf]
THETA           [8-9A-Fa-f]
ARTIFACT        [4-7C-Fc-f]
REM             [8-9A-Fa-f]
OREM            [4-7C-Fc-f]
PREM            [2367ABEFabef]
QREM            [13579BDFbdf]
alphadata       [0]{ALPHA}{HEXNUM}{HEXNUM}
notalpha        [0]{NOALPHA}{HEXNUM}{HEXNUM}
alphamin        [1]{ALPHA}{HEXNUM}{HEXNUM}
notalphamin     [1]{NOALPHA}{HEXNUM}{HEXNUM}

%%
.* \n           {
                linenum += 1;
                if(listing)
                printf("[%05d]%s",linenum,yytext);
                REJECT;
                }
{alphadata}     {
                alphatime++;
                return(1);
                }
{notalpha}      {
                return(2);
                }

{alphamin}      {
                alphatime++;
                minute++;
                return(3);
                }
{notalphamin}   {
                minute++;
                return(4);
                }
" "     ; /* white space, ignore */
\t      ; /* white space, ignore */
\n      ; /* and count the number of columns */
.       {
         printf("Scanner error in line %d:");
        printf(" Illegal Character:",linenum);
        if(yytext[0]>=' ')
                printf("%c0,yytext[0]);
```

```
        else
                printf("ascii %03o0,yytext[0]);
        }
```

*** alpha.gram.g ***

```
%terminals
        alphadata       1
        notalpha        2
        alphamin        3
        notalphamin     4
%endmarker
        0
%grammar
data    :
        data1           #5
        ;
data1   :
        data2 data1
        |null
        ;
data2   :
        alphadata       #1
        |notalpha       #2
        |alphamin       #3
        |notalphamin    #4
        ;
null    :
        ;
```

*** alpha.parser.c ***

```
#include <stdio.h>
#include "ptables.h"

extern int trace, linenum;
extern char yytext[];

#define ERROR 0
#define SHIFT 1
#define REDUCE 2
#define ACCEPT 3

static gotof(st,sym)
register int st;
int sym;
{
        register int m,n;
```

```
        for(m=gotop[sym-numterms],
            n=gotop[sym-numterms+1]-2;m<n;m+=2) {
                if(gotov[m]==st)
                        return(gotov[m+1]);
        }
        return(gotov[n+1]);
}

static actionf(st,sym)
short st;
register short sym;
{
        register int m,n;

        for(m=actionp[st],n=actionp[st+1]-2;
        m<=n;m+=2) {
                if(actionv[m]==sym)
                        return(actionv[m+1]);
        }
        return(actionv[n+1]);
}

#define SIZE_STATE      16000
#define SIZE_TRACE      16000

int     st_stack[SIZE_STATE];
int     tr_stack[SIZE_TRACE];
int     stptr =0;       /*state stack pointer*/
int     trptr =0;       /*trace stack pointer*/


static pspop(n)
int n;
{
        stptr -=n;
}

static pstop()
{
        return(st_stack[stptr-1]);
}

static pspush(n)
int n;
{
        st_stack[stptr] =n;
        stptr +=1;
}

static  shift_trace(token)
short   token;
{
        tr_stack[trptr] =token;
        trptr +=1;
```

```
        dump_trace();
}

static  reduce_trace(n,nonterm)
short   n,nonterm;
{
        trptr -=n;
        tr_stack[trptr] =nonterm;
        trptr +=1;
        dump_trace();
}

static  dump_trace()
{
        int     char_sum, i;

        char_sum =0;
        for (i =0; i <trptr; i++){
                if (char_sum +strlen
                (snames[tr_stack[i]]) >=80){
                printf("0");
                char_sum =10;
                }
        printf(" %s ", snames[tr_stack[i]]);
        char_sum +=strlen(snames[tr_stack[i]]) +2;
        }
        printf("0);
}

static tracelex(token)
short(token);
{
        printf("    NEW TOKEN    %s");
        printf("    INSTANCE    %s0,snames[token],
              yytext);
}

parser() {
        register int token, act;

        pspush(0);
        token=tnmap[yylex()];
        while(1) {
                act = actionf(pstop(),token);

                switch(act%4) {

                case ACCEPT:
                        return(0);
                case ERROR:
                 printf("#%d syntax error: at");
                 printf(" `%s'0,linenum,yytext);
                 return(1);
                case SHIFT:
```

```
                pspush(act/4);
                if (trace ==1) shift_trace(token);
                token=tnmap[yylex()];
                if (trace ==1) tracelex(token);
                break;
               case REDUCE:
                semaction(prsem[act/4]);
                pspop(prlen[act/4]);
                pspush(gotof(pstop(),prlhs[act/4]));
                if (trace ==1)
                        reduce_trace(prlen[act/4],
                                        prlhs[act/4]);
                        break;
               }
        }
}
```

*** alpha.ptables.c ***

```
extern short numsyms;
extern short numprods;
extern short numterms;
extern short numnterms;
extern short numstates;
extern short tnmap[];
extern short prlhs[];
extern short prlen[];
extern short prsem[];
extern char * snames[];
extern int gotop[];
extern short gotov[];
extern int actionp[];
extern short actionv[];
```

*** alpha.main.c ***

```
#include <stdio.h>

int listing=0, trace=0;

FILE *objf;
static char *objname="asm.inp";

main(argc,argv)
int argc;
char *argv[];
{
        int i;
```

```
        for(i=1;i<argc&&argv[i][0]=='-';i++) {
                switch(argv[i][1]) {
                case 'l':
                        listing=1;
                        break;
                case 't':
                        trace=1;
                        break;
                case 'o':
                        objname=argv[++i];
                        break;
                case 'd':
                        objname=0;
                        break;
                }
        }

        if(i<argc) {
                fclose(stdin);
                if(fopen(argv[i],"r")==NULL) {
                printf("%s error: Cannot open %s0);
                printf(argv[0],argv[i]);
                        exit(1);
                }
        }

        parser();
        exit(0);
}
```

*** alpha.semaction.c ***

```
#include <stdio.h>
int nsec=0;

extern int minute;
extern int alphatime;

describe()
{
        if ( alphatime >= 120 )  printf("H");
        else if ( alphatime > 48 ) printf("M");
           else if ( alphatime > 10) printf("L");
                        else printf("N");
        if ( minute % 60 == 0) printf("0);
}
semaction(n)
int n;

{
```

```
        switch(n) {
        case 0:
                break;
        case 1:
                if ( nsec < 4 ) {
                        alphatime += nsec;
                        nsec = 0;
                        }
                break;
        case 2:
                if ( nsec < 4 ) nsec++;
                break;
        case 3:
                if ( nsec <4 ) alphatime += nsec;
                describe();
                alphatime = 0;
                break;
        case 4:
                describe();
                alphatime = 0;
                break;
        case 5:
                printf("0);
                break;
        default:
                printf("*_*_*_* semantic action");
                printf(" %d not handled.0,n);
                break;

        }
}
*** rslow.Makefile ***


files=main.o lex.yy.o parser.o ptables.o semaction.o

rslow: $(files)
        cc -o rslow $(files) -ll
lex.yy.o: lex.yy.c
        cc -c lex.yy.c
lex.yy.c: lexspecs
        lex lexspecs
main.o: main.c
        cc -c main.c
parser.o: parser.c ptables.h
        cc -c parser.c
ptables.o: ptables.c ptables.h
        cc -c ptables.c
semaction.o: semaction.c
        cc -c semaction.c
ptables.c: gram.g
        slrgen -u gram.g ptables.c
```

*** rslow.lexspecs ***

```
%{
#include <stdio.h>
int linenum=0;
int datanum=0;
int rslowtime=0;
int rslowrecord=0;
int deltarecord=0;
int minute=0;

extern int listing;
%}

DIGIT           [01]
HEXNUM          [0-9A-Fa-f]
ALPHA           [89A-Fa-f]
BETA            [4-7C-Fc-f]
DELTA           [2367ABEFabef]
NODELTA         [014589CDcd]
SIGMA           [13579BDFbdf]
THETA           [8-9A-Fa-f]
ARTIFACT        [4-7C-Fc-f]
REM             [8-9A-Fa-f]
NOREM           [0-7]
OREM            [4-7C-Fc-f]
PREM            [2367ABEFabef]
QREM            [13579BDFbdf]
rslowdata       [0]{HEXNUM}{HEXNUM}{REM}
notrslow        [0]{HEXNUM}{HEXNUM}{NOREM}
rslowmin        [1]{HEXNUM}{HEXNUM}{HEXNUM}
notrslowmin     [1]{HEXNUM}{HEXNUM}{HEXNUM}
delta           [01]{DELTA}{HEXNUM}{HEXNUM}
notdelta        [01]{NODELTA}{HEXNUM}{HEXNUM}

%%
^.*\n           {
                linenum += 1;
                if(listing)
                   printf("[%05d]%s",linenum,yytext);
                REJECT;
                }
{delta}         {
                if (( rslowrecord > 0) && ( rslowtime > 0))
                rslowtime -=1;
                deltarecord = 8;
                REJECT;
                }
{notdelta}      {
                if ( deltarecord > 0) deltarecord -=1;
                REJECT;
```

```
                }
{rslowdata}     {
                rslowtime++;
                rslowrecord = 8;
                return(1);
                }
{notrslow}      {
                if ( rslowrecord > 0) rslowrecord -=1;
                return(2);
                }

{rslowmin}      {
                rslowtime++;
                minute++;
                return(3);
                }
{notrslowmin}   {
                minute++;
                return(4);
                }
" "       ;     /* white space, ignore */
\t        ;     /* white space, ignore */
\n        ;     /* and count the number of columns */
.       {
        printf("Scanner error in line %d:");
        printf("Illegal Character:",linenum);
        if(yytext[0]>=' ')
                printf("%c0,yytext[0]);
        else
                printf("ascii %03o0,yytext[0]);
        }
```

*** rslow.gram.g ***

```
%terminals
        rslowdata       1
        notrslow        2
        rslowmin        3
        notrslowmin     4
%endmarker
        0
%grammar
data    :
        data1           #5
        ;
data1   :
        data2 data1
        |null
        ;
data2   :
        rslowdata       #1
        |notrslow       #2
```

```
        |rslowmin       #3
        |notrslowmin    #4
        ;
null    :
        ;
```

*** rslow.semaction.c ***

```
#include <stdio.h>
int nsec=0;

extern int minute;
extern int rslowtime;

describe()
{
        if ( rslowtime >= 16 )  printf("H");
        else if ( rslowtime > 8 ) printf("M");
                else if ( rslowtime > 1)
                                printf("L");
                        else printf("N");
        if ( minute % 60 == 0) printf("0");
}
semaction(n)
int n;

{
        switch(n) {
        case 0:
                break;
        case 1:
                if ( nsec < 3 ) {
                        rslowtime += nsec;
                        nsec = 0;
                        }
                break;
        case 2:
                if ( nsec < 3 ) nsec++;
                break;
        case 3:
                if ( nsec < 3 ) rslowtime += nsec;
                describe();
                rslowtime = 0;
                break;
        case 4:
                describe();
                rslowtime = 0;
                break;
        case 5:
                printf("0);
                break;
```

```
        default:
                printf("*_*_*_* semantic action");
                printf(" %d not handled.0,n);
                break;

        }
}
```

REFERENCES

Ag67 H.W. Agnew, Jr., J.C. Parker, W.B. Webb, and R.L. William, "Amplitude Measurement of the Sleep Electroencephalogram," Electroenceph. Clin. Neurophysiol., Vol. 22, 1967, 84-89.

Ag73 H.W. Agnew, Jr., "Integrater Analysis of the Sleep Electroencephalogram," Electroenceph. Clin. Neurophysiol., Vol. 34, 1973, 391-397.

Ak74 H. Akaike, "A New Look at the Statistical Model Identification," IEEE Trans. Autom. Control, Vol. AC-19, Dec. 1974, 716-723.

An68 P. Andersen and S.A. Anderson, "Physiological Basis of the Alpha Rhythm," Appleton Publishing Co., New York, 1968.

Au81 J.I. Aunon and D.G. Childers, "Signal Processing in Evoked Potential Research: Averaging and Modeling," Critical Reviews in Bioengineering, Vol. 5, No 4, July 1981, 323-367.

Az82 K. Azumi and S. Sirakawa, "Characteristics of Spindle Activity and Their use in Evaluation of Hypnotics," Sleep, Vol. 5, No. 1, 1982, 95-105.

Ba79 S.S. Barlow, "Computerized Clinical Electroencephalography in Perspective", IEEE Trans. Biomed. Engg., Vol. 26, No. 7, July 1979, 377-391.

Ba82a A. Barr and E.A. Feigenbaum, "The Handbook of Artificial Intelligence, Vol. 1," William Kaufman Inc., Los Altos, California, 1982, 141-222.

Ba82b A. Barr and E.A. Feigenbaum, "The Handbook of Artificial Intelligence, Vol. 2," William Kaufman Inc., Los Altos, California, 1982, 175-222.

Ba53 M.S. Bartlett, "An Introduction to Stochastic Processes with Special Reference to Methods and Applications," Cambridge University Press, New York, 1953.

Ba78 S.C. Bass, J.W. Grundman, and S.E. Belter, "DINAP: A Digital Filter Analysis Program," Program Manual, School of Engineering, Purdue University, Indiana, Mar. 1978.

Bi73 R.G. Bickford, J. Brimm, L. Berger, and M. Aung, "Application of Compressed Spectral Array in Clinical EEG," In: Automation of Clinical Electroencephalography, Ed. by P. Kellaway and J. Peterson, Raven Press, New York, 55-64.

Bi78a C.D. Binnie, B.G. Batchelor, P.A. Bowring, C.E. Darby, L. Herbert, D.S.L. Lloyd, D.M. Smith, G.F. Smith, and M. Smith, "Computer-Assisted Interpretation of Clinical EEGs," Electroenceph. Clin. Neurophysiol., Vol. 44, 1978, 575-585.

Bi78b W.P. Birkemeier, A.B. Fontaine, G.G. Celesia, and K.M. Ma, "Pattern Recognition Techniques for the Detection of Epileptic Transients in EEG," IEEE Trans. Biomed. Engg., Vol. BME-25, No. 3, May 1978, 213-217.

Bl58 R.B. Blackman, and J.S. Tukey, "The Measurement of Power Spectra," Dover Publishing Company, New York, 1958.

Bl74 N.M. Blackman, "Sinusoids versus Walsh Functions," Proc. IEEE, Vol. 62, No. 3, Mar. 1974, 346-354.

Bo71 A. van den Bos, "Alternative Interpretation of Maximum Entropy Spectral Analysis," IEEE Trans. Inform. Theory, Vol. IT-17, July 1971, 493-494.

Bo77 G. Bodenstein ,and H.M. Praetorius, "Feature Extraction from the Electroencephalogram by Adaptive Segmentation," Proc. IEEE, Vol. 65, No. 5, May 1977, 642-652.

Bo78 S.R. Bourne, "An Introduction to the UNIX Shell," In UNIX Programmer's Manual Vol. 2, Bell Telephone Laboratories, Inc., Murray Hill, New Jersey, Jan. 1979.

Bo80 J.R. Bourne, B. Hamel, D. Gise, G.M. Woyce, P.L. Lawrence, J.W. Ward, and P.E. Teschan, "The EEG Analysis System of the National Cooperative Dialysis Study," IEEE Trans. Biomed. Engg., Vol. BME-27, No. 11, Nov. 1980, 656-664.

Bo83 S.R. Bourne, "The UNIX System," Addison-Wesely Publishing Co. Reading, Massachusetts., 1983.

Br75 M.A.B. Brazier, P.H. Crandall. W.J. Brown, "Long Term Follow-up of EEG Changes Following Therapeutic Surgery in Epilepsy," Electroenceph. Clin. Neurophysiol., Vol. 38, 1975, 495-511.

Br80 R. Brooks and J. Heiser, "Some Experience with Transferring the MYCIN System to a New Domain," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 5, Sept. 1980, 477-478.

Bu72 J.P. Burg, "The Relationship Between Maximum Entropy Spectra and Maximum Likelyhood Spectra," Geophysics, Vol.37, April 1972, 375-376.

Ca67 J. Capon, R.J. Greenfield, and R.J. Kolker, "Multidimensional Maximum-Likelyhood Processing of a Large Aperature Seismic Array," Proc. IEEE, Vol. 55, No. 2, Feb. 1967, 192-213.

Ca69 J. Capon, "High-Resolution Frequency-Wavenumber Spectrum Analysis," Proc. IEEE, Vol. 57, Aug. 1969, 1408-1418.

Ch73 D.G. Childers, W.M. Mesa, and O.S. Halpeny, "A Neuronal Population Model for and Simulation of Spatio-Temporal Evoked EEG," IEEE Trans., Syst., Man, and Cyber., Vol. SMC-3, No. 4, July 1973, 336-348.

Ch77 D.G. Childers, "Evoked Response: Electrogenesis, Models, Methodology and Wavefront Recognition and Tracking Analysis," Proc. IEEE, Vol. 65, No. 5, May 1977, 611-626.

Ch81 D.G. Childers and J.I. Aunon, "Spectral Analysis: Prediction and Extrapolation," Critical Reviews in Bioengineering. Vol. 6, Issue 2, Sept. 1981, 133-175.

Co77 B.A. Cohen, and A. Sances, "Stationarity of the Human Electroencephalogram," Med. & Biol. & Computers, Vol. 15, 1977, 513-518.

Co81 P.R. Cohen, and E.A. Feigenbum, "The Handbook of Artificial Intelligence, Vol. 3, " HeurisTech Press, Stanford, California, 1981.

Co65 J.W. Cooley and J.S. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, 1965, 297-301.

Co67 J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "Historical Notes on the Fast Fourier Transform," IEEE Trans. Audio Electroacoust., Vol. AU-15, June 1967, 76-79.

Co80 R. Cooper, J. W. Osselton, J. C. Shaw, "EEG Technology," London, England, Butterworths & Co. Ltd., 1980.

Da82 R. Davis, "Expert System: Where Are We? And When Do We Go From Here?," The AI Magine, Vol. 3, No. 2, Spring 1982, 135-173.

De77 J. Demartine, and A.V. Carrefour, "Topics on Pattern Recognition," In: A Didatic Review of Methods and Applications of EEG Data Processing, Ed. by A. Remond, Elsevier Scientific Publishing Co., New York, 1977, 107-126.

Du73 G. Dumermuth, and T. Keller, "EEG Spectra Analysis by Means of FFT," In: Automation of Clinical Electroencephalography, Raven Press, New York, 1973, 145-160.

Dz84 R. Dzwonczyk, M.B. Howie, and J.S. McDonald, "A Comparison Between Walsh and Fourier Analysis of the Electroencephalogram for Tracking the Effects of Anesthesia," IEEE Trans. Biomed. Engg., Vol. BME-31, No. 8, Aug. 1984, 551-556.

El67 R. Elul, "Gaussian Behavior of teh EEG: Changes During Performance of Mental Task," Science, No. 164, 1976, 328-331.

Er80 L.D. Erman, F.H. Hayes-Roth, V. R. Lesser, and D.R. Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," Computing Surveys, Vol. 12, No. 2, June 1980, 213-253.

Fe71 P.B.C. Fenwick, P. Hiche, J. Dollimore, and G.W. Fenton, "Mathematical Simulation of the Electroencephalogram Using an Autoregressive Series," Bio-Medical Computing, Vol. 2, 1971, 281-307.

Fe77 E.A. Feigenbaum, "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," In: IJCAI 5, 1977, 1014-1029.

Fo82 J.D. Foley, and A. van Dam, "Fundamentals of Interactive Computer Graphics," Addison-Wesley Publishing Company, Reading, Massaccusetts, 1982, 553-574.

Fr69 J.D. Frost, Jr., "Wavelength Analysis of the EEG-The Alpha Profile," Electroenceph. Clin. Neurophysiol., Vol. 27, 1969, 702-703.

Fr73 J.D. Frost, Jr., C.E. Hillman, Jr. and P. Kellaway, "Automatic Interpretation of EEG: Analysis of Background Activity," Computers and Biomedical Research, Vol. 13, 1973, 242-257.

Ga73 J.M. Gaillard and R. Tissot, "Principles of Automatic Analysis of Sleep Records with a Hybrid System," Computers and Biomedical Research, 6, 1973, 1-13.

Ge75 A.S. Gevins, C.L. Yeager, S.L. Diamond, J-P. Spire, G.M. Zeitlin, and A.H. Gevins, "Automated Analysis of the Electrical Activity of the Human Brane (EEG): A progress report," Proceedings, IEEE, Vol. 63, No. 10, Oct. 1975, 1382-1399.

Ge80 A.S. Gevins, "Pattern Recognition of Human Brain Electrical Potentials," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 5, Sept. 1980, 383-404.

Ge83 W.B. Gevarter, "Expert Systems: Limited But Powerful," IEEE Spectrum, Vol. 20, No. 8, Aug. 1983, 39-44.

Gi79 D.A. Giese, and J.R. Bourne, "Syntactic Analysis of the Electroencephalogram," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-9, No. 8, Aug. 1979, 429-435.

Go74 A.R. Gondeck, and J.R. Smith, "Dynamic of Human Sleep Sigma Spindles," Electroenceph. Clin. Neurophysiol., Vol. 37, 1974, 293-297.

Ha75 O.S. Halpeny and D.G. Childers, "Composite Wavefront Decomposition via Multidimensional Digital Filtering of Array Data," IEEE Trans. Circuits and Syst., Vol. CAS-22, June 1975, 552-562.

Ha83a J. Hasan, "Differentiation of Normal and Disturbed Sleep by Automatic Analysis," Turku, Finland, ACTA PHYSIOLOGICA SCANDINVICA Supplementum 526, 1983.

Ha83b F. Hayes-Roth, D.A. Waterman, and D.B. Lenat, "Building Expert Systems," Addison-Wesley Publishing Co. Inc., Reading, Massachusetts, 1983.

Hj70 B. Hjorth, "EEG Analysis Based on Time Domain Properties," Electroenceph. Clin. Neurophsiol., Vol. 29, 1970, 306-310.

Ho79 J.E. Hopcroft, and J.D. Ullman, "Introduction to Automata Theory, Languages, and Computation," Addison-Wesley Publishing Company, Reading, Massachusetts, 1979, 234-260.

Is75a A. Isaksson and A. Wennberg, "Visual Evaluation and Computer Analysis of the EEG: A Comparison," Electroenceph. Clin. Neurophysiol., Vol. 38, 1979. 79-86.

Is75b A. Isaksson and A. Wennberg, "AN EEG simulator - A Means of Objective Clinical Interpretation of EEG," Electroenceph. Clin. Neurophysiol., Vol. 39, 1979, 313-320.

It69 T.M. Itil, D.M. Shapiro, M. Fink, and B.A. Kassebaum, "Digital Computer Classification of EEG Sleep Stages," Electroenceph. Clin. Neurophysiol., Vol. 27, 1969, 76-83.

Ja81 B.H. Jansen, J.R. Bourne, and J.W. Ward, "Spectral Decomposition of EEG Intervals Using Walsh and Fourier Transform," IEEE Trans. Biomed. Engg., Vol. BME-28, No. 12, Dec. 1981, 836-838.

Jo69 J.P. Joseph, A.Remond, H. Rieger, and N. Lesevere, "The Alpha Average. II. Quantitative Study and the Proposition of a Theoretical Model," Electroenceph. Clin. Neurophysiol., Vol. 26, 1969, 350-360.

Jo75 S.C. Johnson, "YACC: Yet Another Compiler-Compiler," Comp. Sci. Tech. Rep. No. 32, Bell Laboratories, Murray Hill, New Jersey, July 1975.

Jo76 L.C. Johnson, K. Hasan, R.G. Bickford, "Effect of Flurazepam on Sleep Spindle and K-complexes," Electroenceph. Clin. Neurophysiol., Vol. 40, 1976, 67-77.

Ka73 N. Kawabata, "A Nonstationarity Analysis of the Electroencephalogram," IEEE Trans. Biomed. Engg., Vol. BME-20, 1973, 444-452.

Kt81 P.K. Ktonas, and A.P. Gosalia, "Spectral Analysis vs. Period-Amplitude Analysis of Narrowband EEG Activity: A comparison Based on The Sleep Delta-Frequency Band," Sleep, Vol. 4, No. 2, 1981, 193-206.

Ku80 C.A. Kulikowski, "Artificial Intelligence Methods and Systems for Medical Consultation," IEEE Trans. PAMI, Vol. PAMI-2, No. 5, Sept. 1980, 464-476.

La70 L.E. Larson, and D.O. Walter, "On Automatic Methods of Sleep Staging by EEG Spectra," Electroenceph. Clin. Neurophysiol., Vol. 28, 1970, 459-467.

La76 H. Larsen, "An Algorithm to Compute the Sequency-ordered Walsh Transform," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-24, Aug. 1976, 334-335.

La80 H. Larsen, and D.C. Lai, "Walsh Spectral Estimates with Applications to the Classification of EEG Signals," IEEE Trans. Biomed. Engg., Vol. BME-27, No. 9, Sept. 1980, 485-492.

Le75 M.E. Lesk and E. Schmidt, "LEX-A Lexical Analyzer Generator," Comp. Sci. Tech. Rep. No. 39 Bell Laboratories, Murray Hill, New Jersey, Oct. 1975.

Lo84 G. Logothetis, "A Table Generator for SLR(1) Parsers," In preparation. 1985.

Ly77 P.A. Lynn, "On Line Digital Filters for Biological Signals: Some Fast Design for a Small Computer," Medical & Biological Engg. & Computing, Vol. 15, Sept. 1977, 534-540.

Ma72 W.B. Martin, L.C. Johnson, S.S. Viglione, P. Naitoh, R.D. Joseph, and J.D. Moses, "Pattern Recognition of EEG-EOG as a Technique for All Night Sleep Scoring," Electroenceph. Clin. Neurophysiol., Vol. 32, 1972, 417-427.

Ma73 M. Matousek, and I. Peterson, "Frequency Analysis; Processing of Data," Handbook of EEG Clinical Neurophysiol., 5A, 1973, 61-66.

Ma76 J. Makhoul, "Linear Prediction: A Tutorial Review," Proc. IEEE, Vol. 63, April 1975, 561-580.

Mc75 J.A. McEwen, and G.B. Andersen, "Modeling of Stationarity and Gaussianity of Spontaneous EEG Activity," IEEE Trans. Biomed. Engg., Vol. BME-22, 1975, 361-368.

Mc77 C.D. McGillem and J.I. Aunon, "Measurements of Signal Components in Sigle Visually Evoked Brain Potentials," IEEE Trans. Biomed. Engg., Vol. BME-24, May 1977,232-241.

Mc81 C.D. McGillem, J.I. Aunon, and D.G. Childers, "Signal Processing in Evoked Potential Research: Applications of Filtering and Pattern Recognition," Critical Reviews in Bioengineering, Vol. 6, No. 3, Oct. 1981, 225-265.

Mc82 J. McDermott, "R1: A Rule-Based Configurer of Computer Systems," Artificial Intelligence, Vol. 19, 1982, 39-88.

Me81 V. Melle, "System Aids in Constructing Consultation Programs," UMI Research Press, Ann Arbor, Michigan, 1981.

Mi79 D. Michael and J. Houchin, "Automatic EEG analysis: A Segmentation Procedure Based on the Autocorrelation Function," Electroenceph. Clin. Neurophysiol., Vol. 46, 232-235.

Mi75 M. Minsky, "A Framework for Representing Knowledge," In the Psychology of Computer Vision, P.H. Winston (Ed), McGrow-Hill, New York, 211-277, 1975.

Mi81 D. Michie, "Expert Systems," The Computer Journal, Vol. 23, No. 4, 1981, 369-376.

Mi82 R.A. Miller, H.E. Pople, Jr, and J.D. Myers, "INTERNIST-I, An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine," The New England Journal of Medicine, Vol. 307, No. 8, Aug. 19, 1982, 468-476.

Ne69 A. Newell, and H. Simon, "Human Problem Solving," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

Ni80 N. Nilson, "Principles of Artificial Intelligence," Palo Alto, California, Tioga Publishing Co., 1980, 17-49.

Nu81a Paul L. Nunez, "Electric Fields of the Brain," Oxford University Press, Oxford, New York, 1981, 349-400.

Nu81b Paul L. Nunez, "A study of origins of the time dependenvies of scalp EEG: 1-Theoretical Basis," IEEE Trans. Bio. Engg. Vol. BME-28, No. 3, March 1981, 271-280.

Op75 A.V. Oppenheim, and R.W. Schafer, "Digital Signal Processing," Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.

Pa76 S.G. Pauker, G.A. Gorry, J.P. Kassirer, and W.B. Schwartz, "Towards the Simulation of Clinical Cognition, Taking a Present Illness by Computer," The American Journal of Medicine, Vol. 60, June 1976, 981-996.

Pe77 D. Perry, "Linear Distortion," Journal of Audio Engineering Society, Vol. 24, No. 5, June 1976, 346-367.

Pr77 H. M. Praetouries, G. Bodenstein and O. D. Creutzfeldt, "Adaptive segmentation of EEG records: A new approach to automatic EEG analysis," Electroenceph. Clin. Neurophysiol., Vol. 42, 84-94.

Pr79 J.C. Principe, J.R. Smith, S.K. Balakrishanan, and Arnold Pagie, " Microcomputer-based Digital Filters for EEG Processing," IEEE Trans. ASSP, Vol. 27, No. 6, Dec. 1979, 697-705.

Pr82 J.C. Principe and J.R. Smith, "Sleep Spindle Characteristics as a Function of Age," Sleep, Vol. 5, No. 1, 1982, 73-84.

Pr85 J.C. Principie J.R. Smith, "Design and Implementation of Linear Phase Filters in a Microcomputer System," In preparation. 1985.

Re68 A. Remond, "The Importance of Topographic Data in EEG Phenomena and an Electric Model to Reproduce Them," Electroenceph. Clin. Neurophysiol., Suppl. 27, 1968, 27-49.

Re77 A. Remond, "EEG Informatics. A didactic review of methods and applications of data processing," Elsevier Scientific Publishing Co., New York, 1977.

Ri83 E. Rich, "Artificial Intelligence," McGraw-Hill, New York, 1983.

Ro70 R. Roessler, R. Collins, and K. Ostman, "A Period Analysis Classification of Sleep Stages," Electroenceph. Clin. Neurophysiol., Vol. 29, 1970, 358-362.

Sa71 B. Saltzberg and N.R. Burch, "Period Analytic Estimates of Moments of the Power Spectrum: A Simplified EEG Time Domain Procedure," Electroenceph. Clin. Neurophysiol., Vol. 30, 1971, 568-570.

Sa80 A.C. Sanderson, J. Segen, and E. Richey, "Hierarchical Modeling of EEG Signals," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No.5, Sept. 1980, 405-415.

Se79 R.W. Sencaz, J.I. Aunon, and S.D. MaGillem, "Discrimination among Visual Stimuli by Classification of Their Single Evoked Potentials," Med. & Biol. Eng. & Comput., Vol. 17, 1979,435-443.

Sh71 J.C. Shaw, "An EEG signal generator with variable time delayed output," Med. Biol. Engg, Vol. 9, 71-73.

Sh73 E.H. Shortliffe, S.G. Axline, B.G. Buchanan, T.C. Merigan, and S.N. Cohen, "An Artificial Intelligence Program to Advise Physicians regards Antimicrobial Therapy," Comput. Biomed. Res., Vol. 6, 1973, 544-560.

Sh75 E.H. Shortliffe, "A Model of Inexact Reasoning in Medicine," Mathematical Biosciences, Vol. 23, 1975, 351-379.

Sh76 E.H. Shortliffe, "Computer-Based Clinical Decision Aids: MYCIN," Elsevier, New York, 1976.

Sh79 E.H. Shortliffe, B.G. Buchanan, and E.A. Feigenbaum, "Knowledge Engineering for Medical Decision Making. A Review of Computer-Based Clinical Decision Aids," Proc. IEEE, Vol. 69, Sept. 1979, 1207-1224.

Si76 L.D. Silverstein and C.M. Levy, "The Stability of the Sigma Sleep Spindle," Electroenceph. Clin. Neurophysiol., Vol. 40, 1976, 666-670.

Sm74 J.R. Smith, "Automatic Analysis and Detection of EEG Spikes," IEEE Trans. Biomed. Engg., Vol. BME-21, No. 1, Jan. 1974, 1-7.

Sm75 J.R. Smith, W. F. Funke, W.C. Yeo, and R.A. Ambuehl, "Detection of Human Sleep EEG Waveforms," Electroenceph. Clin. Neurophysiol., Vol. 38, 1975, 435-437.

Sm78 J.R. Smith, "Computers in Sleep Research," Critical Reviews in Bioengineering, Vol. 3, Dec. 1978, 93-148.

Sm80 J.R. Smith, "EEG Waveform Detection," Proceedings, 2nd Annual Conference of the Engineering in Medicine and Biology Society, Sept. 1980, 235-238.

Sm81 W.D. Smith, "Walsh versus Fourier Estimatiors of the EEG Power Spectrum," IEEE Trans. Biomed. Engg., Vol. BME-28, No. 11, Nov. 1981, 790-793.

St82 M. Stefik, J. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth,and E. Sacerdoti, "The Organization of Expert Systems, A Tutorial," Artificial Intelligence, Vol. 18, No. 2, 135-173.

Sz79 P. Szolovits and S.G. Pauker, "Computer and Clinical Decision-Making: Whether, How, and for Whom?," Proc. IEEE, Vol. 67, 1979, 1224-1226.

To74 J.T. Tou, and R.C. Gonzalez, "Pattern Recognition Principles," Addison-Wesely Publishing Co., Reading, Massachusetts, 1974.

To75 H. Tong, "Autoregressive Model Fitting with Noisy Data by Akaike's Information Criterion," IEEE Trans. Inform. Theory, Vol. IT-21, July 1975, 476-480.

To76 H. Tong, "More on Autoregressive Model Fitting with Noisy Data by Akaike's Information Criterion," IEEE Trans. Inform. Theory, Vol. IT-23, May 1977, 409-410.

Ul75 T.J. Ulrich, adn T.N. Bishop, "Maximum Entropy Spectral Analysis and Autoregressive Decomposition," Rev. Geophysics and Space Phys., Vol. 13, Feb. 1975, 183-200.

Vi77 J.J. Vidal, " Real-Time Detection of Brain Events in EEG," Proceedings, IEEE, Vol. 65, No. 5, May 1977, 633-641.

We70 P.C. Welch, "The Use of Fast Fourier Transform for the Estimation of Power Spectra," IEEE Trans. Audio Electroacust., Vol. AU-15, June 1970, 70-73.

We71 A. Wennberg, and L.H. Zetterberg, "Application of a Computer-Based Model for EEG Analysis," Electroenceph. Clin. Neurophysiol., Vol. 31, 1971, 457-468

We72 H. Weinberg and R. Cooper, "The Recognition Index: A Pattern Recognition Technique for Noisy Signals," Electroenceph. Clin. Neurophysiol., Vol. 33, 1972, 608-613.

We84 S.M. Weiss, and C.A. Casimir, "A Practical Guide to Designing Expert System," Rowman & Allanheld Publishers, Totowa, New Jersey, 1984.

Wi79 P.H. Winston, "Artificial Intelligence," Reading, Massachusetts, Addison-Wesely Publishing Co., 1979.

Ye72 W.C. Yeo, and J.R. Smith, "Walsh Power Spectra of Human Electroencephalogram," In Proc. Appl. Walsh Functions, 1972, 159-162.

Yu80 T.P. Yunck, and F.B. Tuter, "Comparison of Decision Rules for Automatic EEG Classification," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 5, Sept. 1980, 420-428.

Ze69 L.H. Zetterburg, "Estimation of Parameters for a Linear Difference Equation with Application to EEG Analysis," Mathematical Biosciences, Vol. 5, 1969, 227-275.

BIOGRAPHICAL SKETCH

Cheoung Nam Lee was born on December 1,1950, in Seoul, Korea. He received his bachelor's degree in electrical engineering from Seoul National University in 1976. After graduation, he worked for Whashin-Sony Co. Ltd. and Gold Star Co. Ltd. for five years. He entered the University of Florida in September, 1980, and received the Master of the Science degree in 1981.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jack R. Smith, Chairman
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Donald G. Childers, Cochairman
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Leon W. Couch
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Arnold H. Nevis
Professor of Electrical
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Ira S. Fischler
Associate Professor of
Psychology

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1985

Hubert A. Bari
Dean, College of Engineering

Dean, Graduate School